UNIVERZITA KARLOVA

Přírodovědecká fakulta

Katedra aplikované geoinformatiky a kartografie

Studijní program: Geografie

Studijní obor: Kartografie a geoinformatika

Bc. Kristýna Měchurová

GENERALIZACE LOD2 MODELŮ BUDOV METODOU AGREGACE

GENERALIZATION OF LOD2 BUILDING MODELS USING THE AGGREGATION METHOD

Diplomová práce

Vedoucí diplomové práce: RNDr. Lukáš Brůha, Ph.D.

Praha, 2020

Prohlášení:

Prohlašuji, že jsem závěrečnou práci zpracovala samostatně a že jsem uvedla všechny použité informační zdroje a literaturu. Tato práce ani její podstatná část nebyla předložena k získání jiného nebo stejného akademického titulu.

V Praze, 5. 5. 2020

_____

Bc. Kristýna Měchurová

Poděkování:

Ráda bych tímto poděkovala vedoucímu mé práce RNDr. Lukáši Brůhovi, Ph.D. za věnovaný čas, cenné rady a odbornou pomoc při zpracování práce. Dále bych chtěla poděkovat Ing. Jiřímu Hutárkovi a Mgr. Tomáši Pokornému za konzultace, cenné rady a pomoc v oblasti programování. V neposlední řadě bych chtěla poděkovat rodině a přátelům za podporu při psaní diplomové práce a během celé doby studia.

Generalizace LOD2 modelů budov metodou agregace

Abstrakt

Práce se zabývá návrhem a implementací postupu agregace 3D modelů budov LOD2. Pomocí metody matematické optimalizace je navržen takový postup, aby bylo dosaženo globálně optimálního řešení. Budovy jsou agregovány na základě podobnostních charakteristik typických pro LOD2, např. typ střechy. Při agregaci je kladen důraz na to, aby byly minimalizovány objemové změny tělesa a zároveň aby se minimalizoval počet agregátů. Je vytvořena optimalizační úloha v podobě skriptu s volitelnými parametry tak, aby mohl skript posloužit široké škále uživatelů. Vstupní data jsou vytvořena pomocí metody procedurálního modelování a následně upravena tak, aby budovy tvořily souvislé bloky. Nakonec je za účelem názornosti výsledků optimalizace navržen a implementován postup vizualizace optimalizační úlohy.

Klíčová slova: 3D GIS, generalizace, agregace, matematická optimalizace, procedurální modelování

Generalization of LOD2 building models using the aggregation method

Abstract

The thesis proposes and implements a method of 3D building models aggregation. The procedure achieves global optima by the means of mathematic optimization. Buildings are aggregated according to similarity characteristics typical for LOD2, e. g. roof type. Aggregation process is driven by minimalization of volume changes and of the aggregate count. The optimization problem was implemented as a Python script with optional parameters to meet custom demands of a wide range of users. Input data models of buildings are created by the method of procedural modelling. Its outcome is further restructured into form of continuous blocks. Finally, the visualization procedure is designed and implemented to illustrate the results of optimized aggregation of 3D building models.

Keywords: 3D GIS, generalization, aggregation, mathematical optimization, procedural modelling

Obsah

Seznam obrázků

Seznam tabulek

Seznam grafu

Seznam zkratek

CityGML	City geography markup language
GML	Geography markup language
IP	Integer programming
LOD	Level of Detail
LP	Linear programming
MIP	Mixed integer programming
OBJ	Wavefront 3D model format
OGC	Open Geospatial Consortium

1. Úvod

V tradiční kartografii jsou objekty znázorňovány dvou-dimenzionálně (2D), tedy v ploše. Touto reprezentací jsou vizuálně zanedbávány informace o výšce a tvaru, případně objemu tělesa na zemském povrchu. Obecně jde o radikální zjednodušení prostorových objektů, tím dochází ke ztrátě důležitých informací a k eliminaci možností dalších analýz souvisejících s třídimenzionálním prostorem (3D), který se více přibližuje světu reálnému. Proto je snahou vytvářet trojrozměrné modely, a tím tak rozšířit možnosti zkoumaných jevů a analýz.

Mezi významné objekty, které nesou řadu informací v 3D prostoru, patří trojrozměrné modely měst a zejména budov. Ty jsou důležité v různých GIS aplikacích, jako jsou například urbanistické, technické, či environmentální studie (Bijecki a kol. 2014a, Bijecki a kol. 2016a). Pro vizualizaci budov, případně trojrozměrných modelů měst na různých úrovních detailu (*LOD – Level of Detail*), je nutné provádět generalizaci. Jedním z důvodů je snížení objemu dat a s tím souvisejících nároků na vizualizační systém. Dále pomáhá generalizace zvýšit efektivitu prostorového vjemu tím, že sníží hustotu informací (Sester 2007).

Napříč různými úrovněmi detailu je uváděn LOD2 jako dostačují pro celou řadu studií např. analýza hluku, výpočet solárního potenciálu střech a modelování stínů (Bijecki 2016b, Boester a kol. 2015). Každá zmíněná analýza vyžaduje pro svoji správnost dodržení určitých charakteristik budovy, jako je objem či výška objektu nebo typ střešní plochy. Při jejich zachování je vhodné agregovat sousední budovy do souvislých bloků a tím snížit výpočetní náročnost modelu.

Mezi existujícími řešeními generalizace 3D modelů budov za použití metody agregace, které řeší celou budovu včetně střechy, lze identifikovat dva hlavní přístupy reprezentované ku příkladu prací Ge (2018), resp. Kada (2007 a 2011). První případ řešení generalizační úlohy vychází z tvaru střech, druhý z otisku budov na reprezentaci terénu. Každý přístup má své výhody a nevýhody. Oba zmíněné přístupy řeší budovu spíše z pohledu geometrie a zabývají se, jakým způsobem agregovat jednotlivé plochy budovy. Neřeší budovu jako celek, který má určité parametry jako je objem a výška, jež ho charakterizují. Motivací této práce je pomoci překlenout uvedenou mezeru zmíněných přístupů.

2. Cíle práce

Proces agregace budov do bloků vyžaduje jako první krok definici sousedství. Tento krok je následován rozhodnutím, jakou metodu použít pro určení, které budovy agregovat. Agregační metody typicky využívají rozličné heuristické přístupy. Příkladem může být metoda rostoucích regionů (region growing; van Oosterom (1995, cit. v Guercke a kol. (2011), s. 3)). Společným rysem těchto metod je, že mohou snadno dosahovat pouze lokálně optimálních výsledků a míjet optima globální. Guercke a kol. (2011) představil metodu globálně optimální agregace LOD1 budov, která je založena na smíšeném celočíselném programování (Mixed Integer Programming (MIP): Dantzig, 1963). Haunert (2008) ukázal na příkladu agregace sousedících ploch, že tento přístup je NP-úplným problémem. Nicméně vzhledem k charakteru městské zástavby, která je cestní sítí přirozeně dělena do bloků, lze snadno rozdělit optimalizační úlohu na dílčí části o nepříliš vysokém počtu budov a zamezit nepřijatelně dlouhému běhu výpočtu. Výhodou MIP je skutečnost, že umožňuje popsat kombinatorické optimalizační problémy, a je-li možné řešený problém zformulovat jako MIP, existují dostupné a velmi efektivní knihovny, které umí tento problém vyřešit bez potřeby implementace vlastního řešení na míru konkrétnímu optimalizačnímu problému.

Hlavním cílem diplomové práce je navrhnout a implementovat postup globálně optimální agregace 3D modelů budov na úrovni LOD2.

Tato práce naváže na dosavadní výsledky prezentované v Guercke a kol.(2011) pro úroveň LOD1. Tyto poznatky adaptuje a rozšíří optimalizační úlohu o parametry střech, které představují hlavní sémantický rozdíl v popisu budov mezi úrovněmi LOD1 a LOD2.

S vědomím skutečnosti, že rozsáhlé modely měst jsou stále častěji tvořeny automatickými metodami zahrnující prvky procedurálního modelování, diplomová práce představí postup tvorby 3D modelů budov pro experimentální část práce právě na tomto principu. Tento postup implementuje tak, že vytvořená data budou vhodným vstupem pro agregační metodu. To znamená, že kromě jejich správnosti (zejména topologické konzistence) budou též v takovém formátu, aby bylo možné přistupovat k jejich geometrii a odvozovat ty charakteristiky budov, které jsou klíčové pro rozhodovací proces agregační metody.

Za účelem názornosti prezentace výsledků optimalizace bude též navržen a implementován postup vizualizace získaných informací o optimálním provedení agregace. Tento krok ovšem nebude zahrnovat další automatizaci zpracování modelů budov v podobě simplifikace obrysu vzniklých bloků a s tím spojenou redukci geometrie (bodů, hran a ploch), která tvořila původní modely budov.

3. Generalizace budov

Kartografická generalizace urbánních dat zahrnuje tyto generalizační operátory: eliminaci, přemístění, agregaci, zvětšení, eliminaci detailů, zarovnání a typifikaci (McMaster, She 1992). Eliminace je vhodná k odstranění malých budov, jako jsou přístřešky pro auta a pergoly. Přemístění je nutné pro topologickou správnost modelu, pokud se při změně měřítka zmenší okolí daného objektu, je možné, že objekt bude zasahovat do přilehlé komunikace. Agregace seskupuje jednotlivé budovy do bloků. Eliminace detailů odstraňuje zanedbatelné prvky, jako jsou malé výstupky obrysů budov atd. Tím se snižuje zaplněnost modelu a ten je přehlednější, což zvyšuje jeho vizuální kvalitu. Zarovnání zachovává pravé úhly mezi jednotlivými částmi budovy. Typifikace zachovává pravidelný vzor ve větší skupině budov tím, že tuto skupinu nahradí skupinou menší se stejným vzorem.

Tato práce bude zaměřena především na nalezení vhodné metody agregace budov v blocích do souvislých celků s ohledem na zachování základních charakteristik budovy jako je např. typ střechy.

Při práci s 3D modely měst je nutné zmínit datový model CityGML (City Geography Markup Language). Jde o OGC (Open Geospatial Consortium) normu, která zavádí model pro budování a sdílení 3D modelů měst. Cílem této normy je zavést definice základních entit, atributů a vztahů v 3D modelu města, za účelem opětovného užití daných dat v různých aplikačních oblastech (Gröger a kol. 2012).

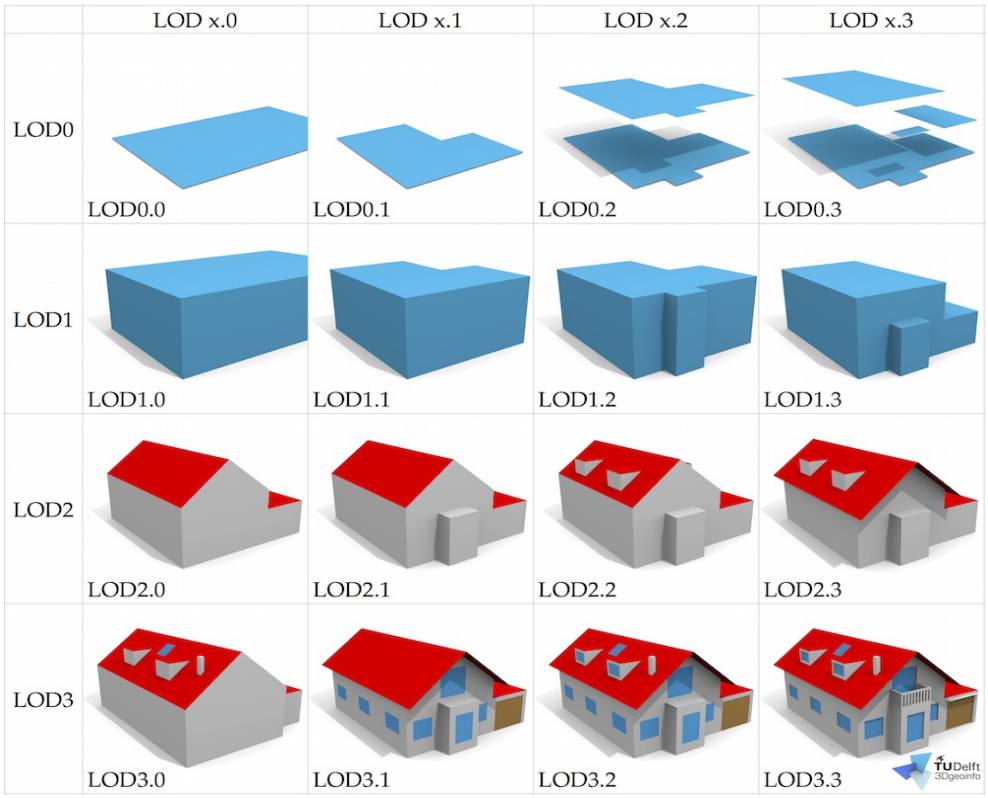
V následujících podkapitolách jsou blíže rozepsány jednotlivá témata: kapitola *3.1* řeší úroveň detailu. Ta je potřeba k definici, jak podrobná data budou agregována. Kapitola *3.2* se zabývá různými pohledy na problém generalizace budov. Podkapitola *3.2.1* řeší obecné přístupy generalizace budov, jejichž součástí je i agregace. Následující kapitola *3.2.2* se zabývá přístupy agregace budov LOD2, avšak většinou pouze pro dvě budovy. Přístupy agregace budov v blocích řeší kapitola *3.2.3*. V následující kapitole *3.3* budou řešeny důležité parametry budovy, které budou vstupovat do programu. Kapitola *3.4* popisuje, co je to matematická optimalizace. V následující kapitole *3.4.1* je již pohlíženo na metodu agregace, jako na optimalizační úlohu lineárního programování. Kapitola uvádí výhody tohoto přístupu, oproti přístupům zmíněných v předchozích kapitolách.

3.1 Úroveň detailu (LOD – Level of Detail)

Úroveň detailu patří mezi nezbytné charakteristiky modelů měst, či jednotlivých budov. Používá se pro definování rozdílných reprezentací reálného světa a charakterizuje, jak podrobný je daný model.

Pojem *Level of Detail* je převzatý z počítačové grafiky, ale jeho chápání v geografických modelech měst je odlišné. V geografické reprezentaci je na rozdíl od počítačové grafiky kladen větší důraz na geometrickou a topologickou přesnost modelů (Bijecki 2017). Úroveň detailu je důležitým pojmem při generalizaci modelů měst. Ta je většinou prováděna z jedné úrovně detailu do druhé.

CityGML jakožto standard pro práci s 3D modely měst je reprezentován pěti rozdílnými úrovněmi detailu (LOD0 – LOD4), kdy podrobnost a přesnost modelů s rostoucím číslem stoupá (Gröger, Plumer 2012). Zatímco LOD0 je definovaný jako půdorys budovy. LOD1 je již trojrozměrný model budov bez střešních ploch. Ty jsou na této úrovni definované jako rovnoběžné plochy s půdorysem. LOD2 rozšiřuje model o různé druhy střešních ploch. LOD3 přidává detaily budovy, těmi jsou komíny, okna, fasáda a dveře. LOD4 pak zahrnuje i interiér budovy. Bijecki a kol. (2016b) ve své práci poukazuje na možnou rozdílnost zobecnění v jednotlivých úrovních detailu a zavádí vylepšení ve formě sady 16 úrovní detailu, viz obrázek č. 1. Čím vyšší stupeň detailu, tím je náročnější pořizovaní dat, modelování a výpočetní náročnost modelu (Bijecki a kol. 2014b a Boester a kol. 2015).



Obrázek č. 1: Rozšíření úrovní detailu
(Převzato z: Biljecki a kol. 2016b)

Tato práce se bude zabývat generalizací modelů LOD2.0. Základním faktorem generalizace bude agregace budov do bloků s ohledem na střešní plochu. Z tohoto důvodu je obtížné začlenit generalizaci z jedné kategorie do druhé, jde o jiný přístup.

3.2 Generalizační přístupy

3.2.1 Způsoby generalizace budov za použití metody agregace

Existující přístupy ke generalizaci budov využívají různé pomocné datové struktury. Lze mezi nimi rozlišit následující čtyři přístupy: (a) triangulací, (b) použitím intervalového stromového grafu, (c) morfologickými operátory a (d) použitím jednoúčelného přístupu. Tyto přístupy řeší generalizaci budov komplexně a jejich součástí je také agregace.

Obecný přístup triangulace představil Ware et al. (1995). Trianguluje se ve volném prostoru mezi budovami. Vybírá se takový trojúhelník, který by mohl být přidán do budov. Trojúhelník je vybrán, pokud je vzdálenost mezi budovami nižší než předem

definovaná vzdálenost, a pak jsou dané budovy agregovány v jednu větší. Využití intervalového stromového grafu bylo zkoumáno v Puttern, Oosterom (1998). Další možností generalizace jejíž součástí je agregace, je metoda morfologických operátorů, která přinesla významné výsledky.

Damen, Kreveld, Spaan (2008) ukazují výhodu použití morfologických operátorů oproti použití stromového intervalového grafu či triangulace. Jedná se o nový přístup vylepšení metody morfologických operátorů, který eliminuje malé otvory a průchody mezi budovami a budovy agreguje do jedné. Navíc experimentální výsledky poukazují na to, že tento přístup přináší lepší výsledky, než při použití stromového intervalového grafu a triangulace. Ačkoliv tento přístup přinesl celou řadu poznatků a řeší vztahy a topologickou správnost mezi budovami, celý výzkum je prováděn v ploše a řeší pouze půdorys budov, nikoliv střešní plochy či fasády.

Forberg, Mayer (2003) využívají metody 3D generalizace zjednodušení a agregace za pomoci morfologických operátorů a zakřivení prostoru. Výzkum je založen na metodě paralelního posunu. V přístupu záleží na prahové hodnotě, která pokud u dvou paralelních ploch klesne pod danou hodnotu, plochy se pohybují tak dlouho k sobě, dokud se nesloučí. Princip paralelního posunutí lze uplatnit pouze na ortogonální objekty. To však budovy (zejména z důvodu střešních ploch) nejsou.

V předchozích výzkumech jsou řešena pouze jednoduchá ortogonální tělesa. Složitější plochy, jako jsou sedlové střechy, jsou nahrazené ortogonální plochou.

3.2.2 Agregace budov včetně střech

Kada (2007) provádí zjednodušení půdorysu pomocí dekompozice na jednotlivé plošky, tedy ve 2D. Následně je dekompozice rozšířena do 3D a jsou zjednodušeny také střešní plochy. Tato metoda však nebere v potaz symetrii střech a napojení jednotlivých typů střech není adekvátní, a to z důvodu rozdílnosti výšky střech. Jednotlivé plošky nemusejí z důvodu rozdílné výšky hřbetu střechy splňovat pravidla, aby byly klasifikovány jako části střechy a pak mezi jednotlivými střechami vznikají mezery a střechy nejsou agregovány do jedné. Z těchto důvodu rozděluje střechy na jednotlivé typy. Následně jsou

eliminovány detaily na střešní ploše a střechy jsou agregovány dle určitých parametrů podle typu střechy.

Ge (2018) se ve své práci zabývá generalizací budov s rozdílnými typy střech z LOD2 na strohý LOD2 (coarser LOD2) až LOD1. Narozdíl od ostatních metod je nejprve zjednodušena střecha a na základě tohoto zjednodušení jsou následně rekonstruovány zbylé části budovy. Nejprve jsou rozlišeny jednotlivé budovy, bloky budov a budovní komplex. Následně jsou automaticky rozpoznávány tři typy střech u jednotlivých budov – věžová, sedlová a pultová. Jedním z hlavních rozpoznávacích pravidel typu střechy je počet polygonů, ze kterých je střecha tvořena. Na jednotlivé typy střech jsou poté aplikovány rozdílné metody generalizace (zarovnání, eliminace detailů) se stejnou prahovou hodnotou. Pokud se vedle sebe nachází dvě budovy se stejným typem střechy a rozdílem výšky střechy nižším, než je prahová hodnota, jsou budovy agregovány.

Výše zmíněné práce neřeší objem generalizované budovy. Výška budov je řešena pouze z hlediska prahové hodnoty, např. pokud je rozdíl výšek menší než prahová hodnota, jsou střechy agregovány. Metody neberou v potaz výsledný objem střechy.

3.2.3 Agregace bloků budov

Kada (2011) rozšiřuje svůj přístup dekompozice budov na jednotlivé plošky o metodu matematické morfologie. Vektorová prezentace je nejprve převedena na rastrovou a následně jsou za pomoci morfologických operátorů agregovány budovy do bloků. Autor poukazuje na problém, který nastává u některých typů střech. Například sedlové střechy mohou být agregovány několika způsoby. Z tohoto důvodu uvádí, že je dobré rozlišit jednotlivé typy střech a podle typu je generalizovat.

Xie J a kol. (2012) přináší novou hybridní metodu pro generalizaci a vizualizaci urbánních dat. V závislosti na pohledu pozorovatele jsou vzdálené budovy agregovány do bloků. Metoda řeší agregaci pomocí půdorysu budovy. Střechy jsou generalizovány na plochy rovnoběžné s půdorysem. Liu a kol. (2017) představil agregaci na základě hierarchického algoritmu, založeného na rozpoznávání textur v blocích budov. Metoda používá jako vstupy půdorys budov a texturu, ne však střešní plochy.

Agregace budov do bloků a generalizace střech jsou řešeny separátně. Pokud jsou generalizovány střechy v blocích budov, jsou na ně aplikovány metody, které ve výsledku nezachovávají základní charakteristiky jednotlivých typů střech, jako je symetrie, či výška střechy.

Z výše uvedených kapitol vyplývá, že problém generalizace, jejíž součástí je agregace budov, byl mnohem detailněji rozebrán na ortogonálních tělesech úrovně detailu LOD1. Z metod, které řeší agregaci budov včetně střešní plochy, lze rozlišit dva přístupy. Ani jeden ovšem neřeší jednotlivé parametry budovy, jako objem či výška budovy. Ty mohou být klíčové v různých analýzách. Výška budovy je důležitá například při analýzách dohlednosti z daného bodu, pro plánování letu, nebo při nutnosti osvícení budov. Objem pak při výpočtu spotřeby tepelné energie dané budovy. Ambicí této práce je vytvořit program, který bude agregovat jednotlivé budovy v bloku do spojitých celků s volitelnými parametry objemové a výškové změny, které bude moci uživatel nastavovat, v závislosti na analýze, do které bude následně model budov vstupovat. Zároveň bude brát program v potaz typ střech dané budovy a orientaci vrcholové linie střechy. Na základě pravidel vycházejících z těchto charakteristik budou budovy agregovány.

3.3 Parametry budovy

Modely budov mohou být využity pro účely navigace, či pro simulace šíření hluku. Mohou sloužit jako vstup pro analýzy v krizovém managementu nebo v otázkách životního prostředí (Guercke a kol. 2011). Dále mohou sloužit jako vstup pro modelování stínů, urbanistické studie, či pro solární potenciál střech, nebo instalaci mobilních antén (Boeters a kol. 2015, Kada 2007). Každá analýza klade důraz na jiné parametry budovy. To teoreticky znamená, že podrobnější modely mohou vstupovat do více analýz. Tyto modely však mají vyšší výpočetní náročnost a je obtížnější je modelovat. Proto je snahou vytvořit model nižší LOD, který bude zachovávat parametry potřebné pro danou analýzu.

Z výše uvedených analýz vyplývají zvolené parametry. Při modelování hluku, bude záležet převážně na výšce a objemu budovy. Pro stíny je také důležitá výška a do jisté míry i typ střešní plochy. Ten je důležitý také při instalaci mobilních antén a při výpočtu solárního potenciálu střech.

Volitelnými parametry tedy budou: výška budovy, objem budovy, typ střešní plochy, orientace vrcholové střešní linie.

3.4 Matematická optimalizace – lineární programování

Lineární programování (*anglicky LP – linear programming)* je jedním z metod matematické optimalizace. Pojem lineární programování nesouvisí s programováním počítačů. Vznikl ve 40. letech minulého století, je převzatý z armádní hantýrky a programování v tomto slova smyslu znamená, nalezení optimálního plánu nějakého rozvrhu nebo činnosti. Následně se úlohy lineárního programování začaly používat v ekonomice a dnes mají hojné zastoupení i v dalších odvětvích, jako je například informatika (Matoušek 2006).

Typickým příkladem úlohy lineárního programování může být sestavení jídelníčku splňujícího všechny výživové hodnoty za minimální cenu finančních nákladů. Dále lze pomocí úlohy lineárního programování zjistit největší přenosovou rychlost v síti. Případně optimalizovat výrobu určité suroviny v továrně v závislosti na poptávce, nákladech na výrobu a skladování v daných měsících v roce.

Cílem lineárního programování je nalezení proměnných (parametrů), pro které bude cílová funkce nabývat maximální, či minimální hodnoty za případných omezujících podmínek. Je tedy optimalizováno řešení rovnice omezené soustavou nerovnic. Lineární funkce, která se má minimalizovat, případně maximalizovat se nazývá účelová či objektivní funkce, lze ji zapsat ve tvaru:

$$c^T x = c_1 x_1 + \cdots + c_n x_n \qquad c \in \mathbb{R}^n$$

Lze také říci, že úloha lineárního programování hledá vektor *c* minimalizující (maximalizující) hodnotu dané funkce mezi všemi vektory.

Nerovnicím se obecně říká omezení nebo omezující podmínky a jejich počet se obecně značí *m*. A lze je vyjádřit v obecném tvaru:

$$Ax \leq b$$

Kde *A* je reálná matice typu *m*n* a $b \in \mathbb{R}^n$ je vektor. Přitom relace $\leq$ platí pro oba vektory *c* a *b*, právě tehdy pokud platí po složkách (Matoušek 2006).

Pro úlohu lineárního programování existuje několik možností řešení. Přípustné řešení splňuje $Ax \leq b$. Optimální řešení je přípustné řešení s maximální hodnotou $c^T x$. Nepřípustná úloha je taková, pro kterou neexistuje přípustné řešení. Naopak neomezená úloha je taková, pro kterou může účelová/objektivní funkce nabývat libovolně velkých hodnot (Sgall, 2018).

V určitých úlohách lineárního programování je vyžadováno pouze celočíselné řešení úlohy. Taková úloha se pak nazývá úlohou celočíselného programování (*IP integer linear programming*) a je definována jako:

$$c^T x = c_1 x_1 + \cdots + c_n x_n \quad c \in \mathbb{Z}^n$$

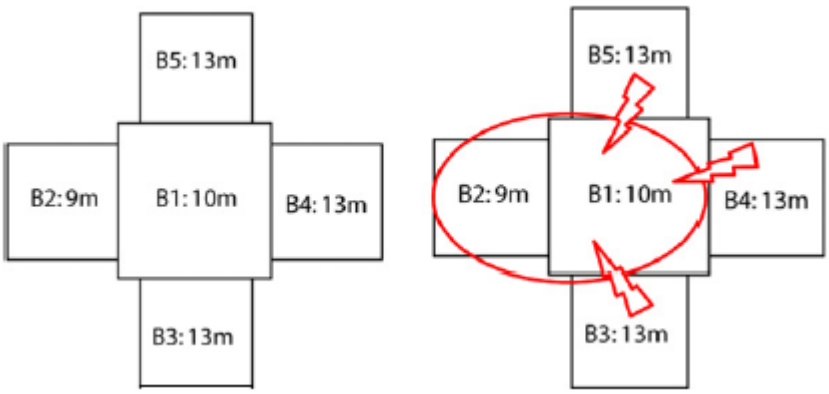
$$Ax \leq b$$

Obdobnou úlohou je úloha smíšeného programování (*MIP – mixed IP*), kdy jsou některé proměnné celočíselné a některé reálné. Úloha binárního programování pak obsahuje binární proměnné (Sgall, 2018).

3.4.1 Lineární programování jako přístup k agregaci budov

Použití matematické optimalizace, přesněji lineárního programování pro řešení agregace budov je výhodné z toho důvodu, že při řádně zvolených podmínkách a správně nastavené objektivní funkci, lze docílit robustního řešení problému. Úloha lineárního programování se snaží ze všech možných řešení najít to nejoptimálnější. Tedy to řešení, při kterém daná funkce nabývá minimálních, případně maximálních hodnot.

Při heuristickém řešení problému je více pravděpodobné, že uživatel při řešení problému zvolí taková omezení, která vynechají optimální řešení. Většina heuristických algoritmů funguje na takovém principu, že je vybrána jedna budova (náhodně nebo na základě určitých pravidel) a jsou nalezeni její sousedé. Následně jsou vybrány budovy, které splňují podmínky agregace – například rozdíl výšek nepřesahuje určitou hodnotu a budovy jsou následně agregovány. V takovém případě se může stát, že heuristický algoritmus nalezne lokální optima, ale vynechá optimum globální (Guercke a kol. 2011).



Obrázek č. 2: Problém heuristických přístupů agregace budov
(Převzato z: Guercke a kol. 2011)

Obrázek č. 2 znázorňuje půdorys bloku budov, který je tvořený pěti budovami. Každá z budov má uprostřed půdorysu definovanou svou výšku. Snahou agregace budov je snížit počet budov tím, že jsou budovy agregovány do jednoho celku a co nejvíce se tak sníží jejich počet ve výsledném modelu. Vždy s ohledem na podmínky agregace. Z obrázku č. 2 vyplývá, že bylo vhodné agregovat budovy B1, B3, B4 a B5 do jednoho celku. U heuristických přístupů mohou nastat dvě nevhodné situace. Pokud bude vybrána jako výchozí budova B2 a agreguje se s budovou B1, bude výška agregátu těchto dvou budov menší než 10. Tím vznikne větší mezera mezi agregátem a dalšími budovami a k agregaci dalších budov nemusí dojít. Druhý nevhodný výsledek agregace může nastat v případě, že je budova B1 zvolena jako výchozí budova a algoritmus má upřednostnit k agregaci takovou budovu, jenž má nejmenší výškový rozdíl s výchozí budovou. Je agregována budova B2 a další budovy se již neagregují (Guercke a kol. 2011).

Oproti heuristickým přístupům vybere úloha lineárního programování ze všech přípustných možností to nejvhodnější (minimální/maximální) a to matematickou cestou.

Agregaci jako optimalizační problém řeší ve své práci Guercke a kol. (2011). Studie je prováděna na budovách LOD1. Jako důležité parametry vstupující do optimalizační úlohy uvádí: výšku budovy s ohledem na výšku terénu a „pseudo objem“ – objem půdorysu. Následně za pomoci omezujících podmínek definuje objektivní funkci a hledá její minima. Snaží se minimalizovat počet agregovaných bloků a zároveň minimalizovat objemové změny agregovaných budov.

Ambicí práce bude rozšířit optimalizační problém na budovy LOD2. Z ortogonálních objektů se stanou budovy se střechami, čímž přibude parametr typu a objemu střešní části.

4. Data a jejich příprava

Kapitola bude věnována datovým zdrojům a vhodnosti použití daných dat pro účely této práce. Následně formálně zdokumentuje jejich generování a metodu úpravy vytvoření bloků budov.

Při výběru dat bylo vycházeno z diplomové práce (Brýdl 2017), konkrétně z kapitoly *Datové zdroje.* Kapitola uvádí několik dostupných 3D modelů měst v různých datových formátech. Více o datových formátech v kapitole *4.1*. Všechny dostupné 3D modely měst jsou však po bližším prozkoumání nevhodné pro další použití. Hlavními nedostatky jsou topologické a geometrické nepřesnosti modelů a z tohoto důvodu jsou nevhodné pro modelování objemového tělesa.

To vedlo k vytvoření vlastního modelu. Jednotlivé budovy byly nejprve vygenerovány pomocí skriptů Random3DCity (Biljecki, Ledoux, Stoter 2016c) ve formátu *CityGML*, viz kapitola *4.2*. Následně byla data upravena do souvislých bloků v programu *SketchUp* a vyexportována do formátů *GML* a *OBJ* viz kapitola *4.3 Úpravy dat a možnosti datových formátů*.

4.1 Datové formáty

Pro práci s 3D soubory existuje řada datových formátů v závislosti na datech a potřebě jejich užití. Mnoho softwarů si pro práci s 3D daty vytváří své vlastní formáty, které nejvíce vyhovují potřebám daného softwaru. Problémem je interoperabilita formátů mezi jinými softwary.

Speciálně pro potřeby modelování měst a městské zástavby vznikl *CityGML* formát, který je definovaný dokumentem *OGC City Geography Markup Language (CityGML) En-coding Standard* pod záštitou *Open Geospatial Consortia (OGC 2012).* Jedná se o úpravu *XML* formátu, který umožňuje hierarchické členění prvků. Formát také podporuje přiřazení popisu jednotlivým plochám budovy, jako je půdorys nebo střecha. Navíc je možné v tomto formátu uložit další informace o budově, například její stáří, či jaký má typ střešní plochy. Nevýhodou *CityGML* formátu je, že ho nepodporuje řada softwarů a jeho velikost je větší, takže vyžaduje dostatek úložného prostoru a je s ním obtížnější pracovat (Biljecki 2015a).

Z tohoto důvodu se někdy data převádí do jiného formátu. Mezi obecné formáty, které lze implementovat do řady programů patří *Wavefront* (tj. *OBJ* soubor). Jde o textový formát, který používají modelační a vizualizační softwary. Na rozdíl od *CityGML* formátu do něho nelze uložit podrobnější informace o daných plochách, zato je snadněji čitelný a zabírá méně úložného prostoru (Biljecki 2015a). Obecně jde tedy o formát, který řeší geometrickou podobu modelu a nelze do něho uložit podrobné dodatečné informace.

CityGML má stromovitou strukturu uložení, z tohoto důvodu je možné jednotlivým budovám přiřadit další informace. Těmi může být stáří budovy, počet pater, typ střešní plochy atd. Navíc jsou jednotlivé plochy tvořící budovu uloženy zvlášť a u plochy je uvedeno, o jaký typ se jedná. Budova LOD 2 má tři typy ploch: půdorys, stěny kolmé na půdorys a střešní plochy. Každá z ploch budovy je uložena zvlášť ve formátu: *LinearRing*. Ten je tvořen všemi vrcholy polygonu, přičemž první a poslední vrchol plochy je stejný.

Formát *Wavefront* má mnohem jednodušší strukturu než *CityGML*. Struktura formátu se liší v závislosti na typu softwaru, ze kterého je formát exportován. Základními prvky *OBJ* formátů této práce jsou tři položky: vrcholy (*vertex*), plochy (*faces*), název objektu (*object name*). Každý vrchol je definován na samostatném řádku. Jednotlivé plochy jsou definované v samostatném řádku a jsou tvořeny indexy vrcholů. Pro každou budovu jsou pak tyto plochy odděleny názvem budovy.

4.2 Procedurální modelování pomocí skriptů Random3Dcity

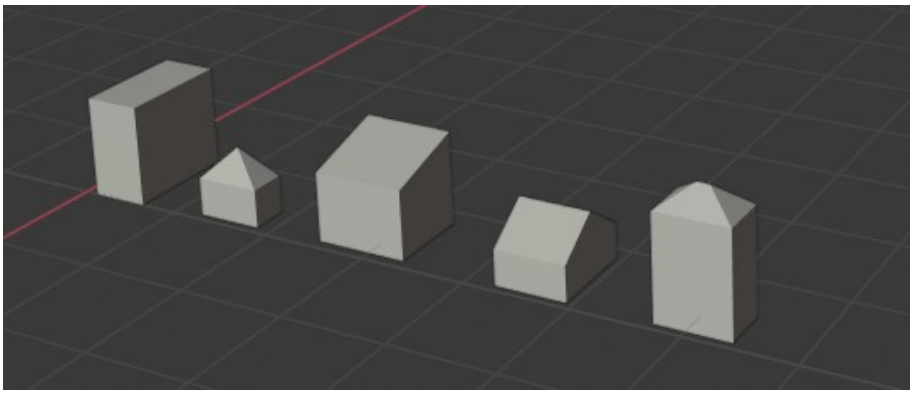
Procedurální modelování je přístup, kterým lze automaticky vytvářet 3D modely. Obecně lze říci, že jde o automatizaci modelování složitých objektů. Ta je založena na generovaní objektů z jeho jednoduchých komponent podle daných pravidel, které jsou závislé na vstupních parametrech (Strachota 2015).

V praxi se často používá pro modelování urbánních prvků jako jsou domy, silnice, značky, parcely, monumenty či celá města. Výhodou procedurálního modelování je poměrně rychlé generování 3D modelů měst, které díky vstupním pravidlům neobsahují topologické chyby (Biljecki, Ledoux, Stoter 2016c). Lze modelovat z reálných vstupních dat, ale také z dat náhodně vygenerovaných. Nevýhodou u modelování reálných měst je

fakt, že výsledný model z důvodu generativní povahy procedurálního modelování nemusí odpovídat realitě (Musialski a kol. 2013).

Sada skriptů pod názvem *Random3DCity* (Biljecki, Ledoux, Stoter 2016c) je v práci použita pro vytvoření náhodné sítě městské zástavby, případně lze s touto sadou skriptů vygenerovat zástavbu z reálných dat, ty však musí podléhat konkrétnímu formátu. Výhodou je, že je výsledný model dostupný ve všech 16 úrovních detailu, viz kapitola *3.1*. Celý balíček je dostupný na stránkách: https://github.com/tudelft3d/Random3Dcity. Obsahuje skript *RandomiseCity.py,* který generuje *XML/GML* soubor s náhodnými parametry budov. Pomocí volitelných parametrů lze výsledný model přizpůsobit potřebám konkrétního výzkumu. Lze měnit natočení, lze generovat silniční síť a vegetaci, možné je také přidat části budovy jako jsou garáže, přístřešky a výklenky. Pomocí drobné úpravy skriptu je možné měnit intervaly náhodných čísel, ze kterých jsou následně tvořeny půdorysy a výšky budov. Skript *GenerateCityGML.py* pak vytváří 16 *GML* souborů charakterizující různé úrovně daného modelu, a navíc čtyři úrovně detailu interiéru.

Takto byla vytvořena data LOD2.0 bez výklenků a natočení. Modely obsahují pět typů střešní plochy: rovnou (*flat*), stanovou (*pyramidal*), pultovou (*shed*), sedlovou (*gabled*) a valbovou (*hipped*), viz obrázek č. 3. Půdorys budov je vždy ortogonální a je tvořen polygonem obdélníkového nebo čtvercového typu ve stejné nadmořské výšce, tedy ve stejné *Z* souřadnici. Model je tvořen v jedné rovině s minimálními odchylkami. Jak již bylo zmíněno výše, budovy v modelu neobsahují žádné výklenky či přidružené budovy jako je garáž, nebo pergola. Hrany půdorysu budov jsou rovnoběžné s lokálním souřadným systémem. Délky stran půdorysu byly nastaveny v rozmezí hodnot $\langle 3{,}10 \rangle$ a výška budovy byla nastavena v rozmezí hodnot $\langle 3{,}20 \rangle$. Vzhledem k tomu, že jde o uměle vytvořený model budov a hodnoty intervalu lze drobnou úpravou změnit, je na uživateli, jakými jednotkami délky a výšky hodnoty označí.



Obrázek č. 3: Typy střech vytvořených modelů
z leva: rovná, stanová, pultová, sedlová, valbová
(Zdroj: vlastní tvorba)

Skript *RandomiseCity.py* vytváří data na pravidelné čtvercové síti, která brání vzniku topologických chyb. V rámci daného čtverce sítě se pak vytváří půdorys budovy menšího obsahu, než je daný čtverec. Z tohoto důvodu nelze jednoduše přizpůsobit skript tak, aby generoval bloky budov. Proto byly vygenerované budovy následně upravovány v dalších programech tak, aby tvořily souvislé bloky.

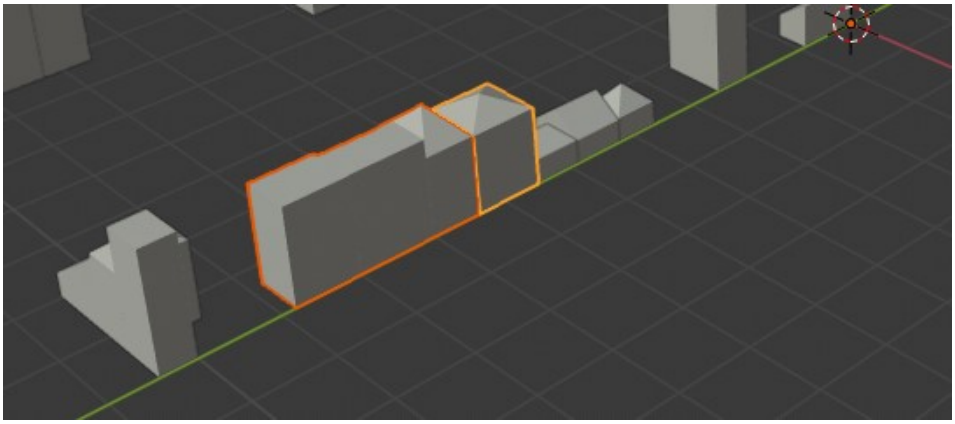
4.3 Úpravy dat a možnosti datových formátů

Při hledání softwaru vhodného pro úpravu dat, bylo zjištěno, že mnoho programů, které se používají pro práci s prostorovými daty nepodporuje *GML* formát souborů. V programech jako je *ArcMap, QGIS, Blender* nešel tento formát otevřít. Vytvořená data byla nakonec importována do programu *SketchUp* za pomoci extenze *CityEditor*. Budovy byly manuálně upraveny tak, aby tvořily souvislé bloky. Byly vytvořeny modely, ve kterých se nachází více bloků budov. I modely, které obsahují pouze jeden blok budov. Takto bylo vytvořeno několik modelů. Následně byly vytvořené modely exportovány do formátu *OBJ* a *GML*.

Nástroj exportu do *GML* formátu extenze *CityEditoru* zanechává všechny původní informace o budově a vytváří nový soubor s příponou *XML* nebo *GML*. Ponechané informace mohou usnadnit další práci. Z výše zmíněného důvodu nekompatibility *GML* formátu s programy pro prostorová data, byla data ještě převáděna do *OBJ* formátu podporovaného více softwary.

Při práci s *OBJ* formáty bylo zjištěno, že existuje velké množství struktur *OBJ* formátů, do kterých lze data importovat. Různé nástroje a programy nabízí rozličné možnosti exportu a struktura vyexportovaných dat je jiná.

Pokud jsou data exportována výchozím nástrojem z programu *SketchUp*, jsou strukturována do takzvaných *Mesh Group* modelů. Nevýhodou tohoto exportu je, že spojuje některé budovy do celků viz obrázek č. 4. Navíc je struktura formátu nepřehledná.



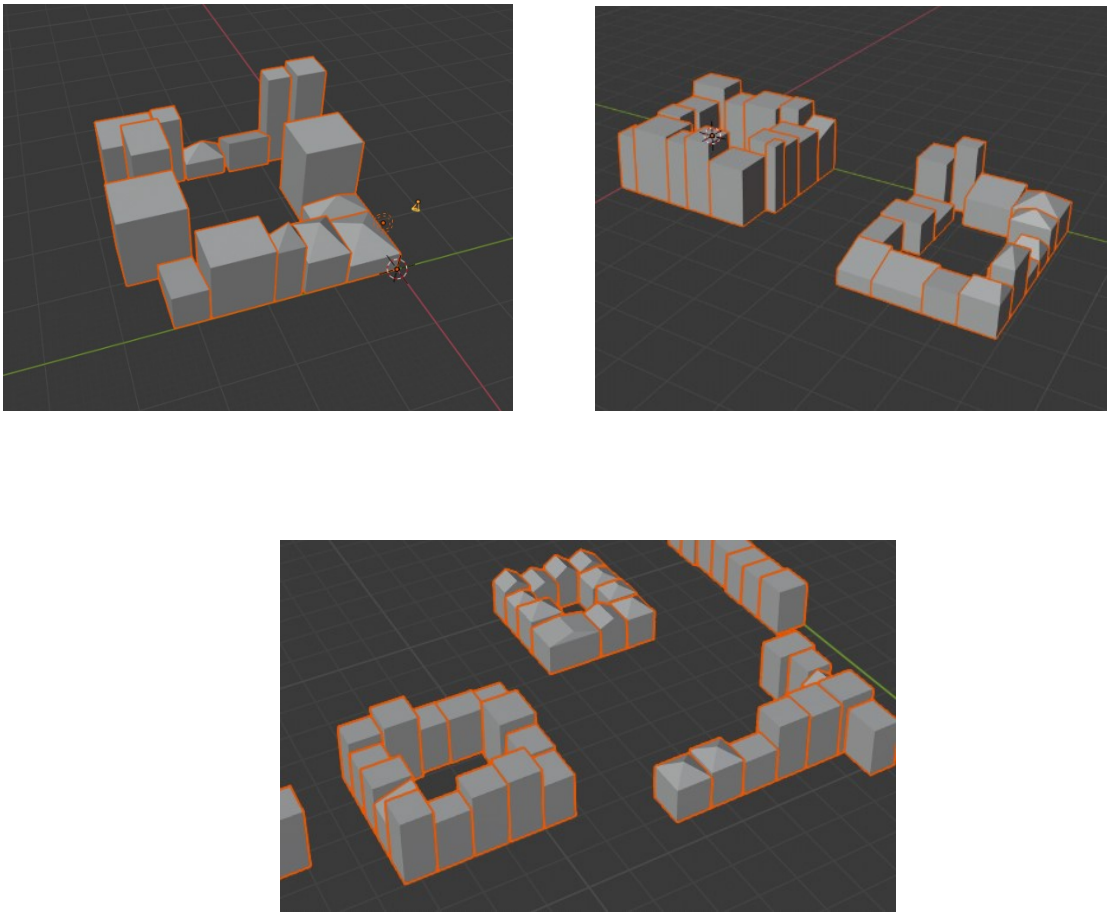
Obrázek č. 4: Budovy spojené po exportu z programu SketchUp
(Zdroj: vlastní tvorba)

Extenze *CityEditor* nabízí také export do formátu *OBJ*. Struktura takto vyexportovaného formátu, není podporována 3D prohlížečem pro Windows ani dalšími softwary a ID jednotlivých budov je navíc rozšířeno o další hodnoty.

Prozatím nejlepším řešením, jak převést data do *OBJ* formátu, je použít sadu skriptů *CityGML2OBJs* navrženou Biljecki, Arroyo, Ohori (2015a). Celý balíček je dostupný na adrese: https://github.com/tudelft3d/CityGML2OBJs. Převod z *GML* do *OBJ* formátu se spustí za pomoci skriptu se stejným názvem, tedy *CityGML2Objs.py*. Pro správné fungování byla nutná úprava skriptu. Konkrétně šlo o smazání argumentu *bAttVal* u funkce *poly_to_obj* na řádku 479. Při ponechání atributu je skript nefunkční. Problémem může být stáří skriptu a s tím související zastaralé, již nepodporované funkce. Skript generuje nejvíce přehlednou strukturu *OBJ* formátu a ponechává ID budov a elementů. Navíc je možné díky skriptu kontrolovat validitu jednotlivých elementů budovy, zvolit translaci souřadnicového systému, triangulovat výsledný model budov nebo generovat jednotlivé části budovy, jako je střecha, půdorys atd. do samostatných *OBJ* souborů.

Nevýhodou všech výše zmíněných metod je, že nekontrolují topologii jednotlivých budov vůči sobě. Buď jsou tyto budovy spojeny do skupin, viz výchozí nástroj *SketchUpu*, nebo jsou převedeny do *OBJ* formátu i s topologickými chybami. To znamená, že jsou zachovány jednotlivé budovy bez změny geometrie i přes to, že se překrývají.

Výslednými daty jsou vytvořené modely bloků budov LOD2.0 ve dvou datových formátech *GML* a *OBJ*. Modely obsahují různé druhy bloků budov, buď jde o bloky tvořící uzavřený celek, nebo o řadu domů vedle sebe. Některé bloky obsahují budovy podobných výšek a stejných typů střechy, jiné se v obou parametrech liší, viz obrázek 5. Hrany půdorysu budov jsou rovnoběžné s lokálním souřadným systémem.



Obrázek č. 5:Ukázka vygenerovaných a upravených modelů bloků budov
(Zdroj: vlastní tvorba)

4.4 Určení základních parametrů budovy

Pro určení základních parametrů jednotlivých budov, jako je jejich výška, objem, typ střechy a počet pater, byl použit datový formát *XML/GML*. Tento formát již obsahuje řadu informací, a tak není potřeba některé informace složitě počítat. Následující podkapitoly řeší čtení *XML/GML* formátu v prostředí programovacího jazyka *Python* viz kap. *4.4.1* a následný výpočet parametrů, které bylo nutné ještě pro optimalizační úlohu vypočítat viz kapitola *4.4.2*. Celý skript je napsaný v programovacím jazyce *Python* ve verzi 2.7.14 a je dostupný na adrese: https://github.com/mechurk/parameters_of_buildings.

4.4.1 Čtení XML/GML formátu v prostředí Python

Pro práci s *XML/GML* formáty byla zvolena knihovna *lxml*. Ta obsahuje řadu funkcí, které usnadňují přístup k jednotlivým položkám v datovém souboru.

Nejprve je nutné definovat *Namespaces*. Datový formát *CityGML* je dostupný ve dvou verzích, 1.0 a 2.0. Přístup k *Namespace* je podmíněný verzí *CityGML* formátu. Pro tuto práci byl použit pouze formát 2.0.

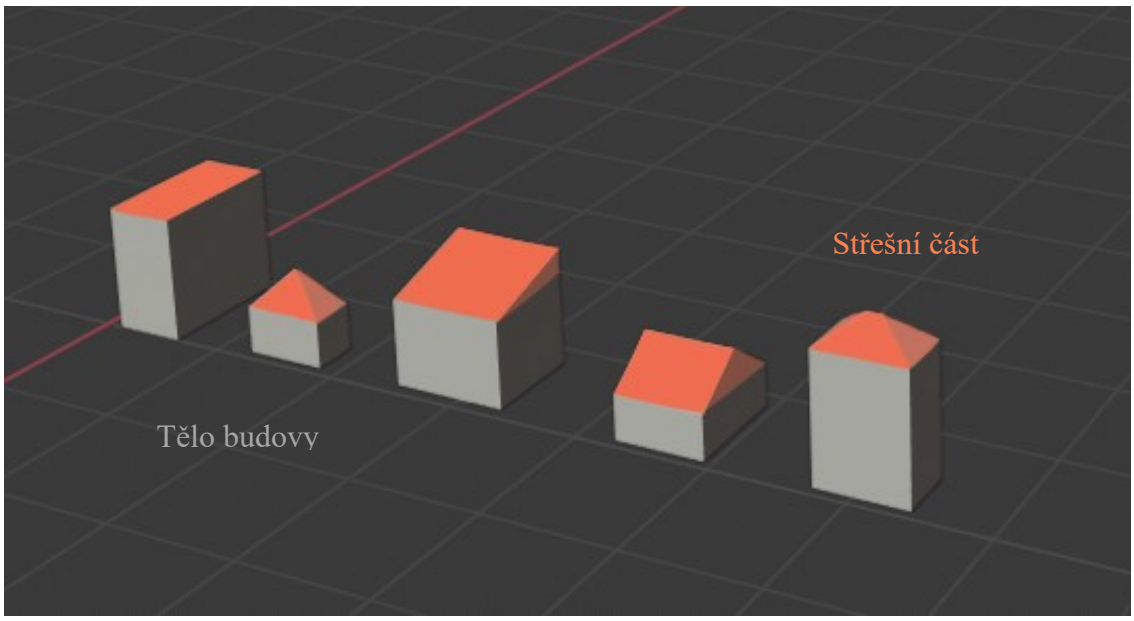
Následně byl do skriptu načten *XML/GML* datový soubor. Byly nalezeny všechny instance položky *cityObjectMember*. Z těchto položek byli vybráni pouze potomci obsahující tag *Building*. Každé budově (potomek obsahující tag *Building*) bylo přečteno její ID, typ střechy, počet pater. Jednotlivé polygony byly rozděleny do zvláštních tříd na základě toho, o jaký polygon se jedná: *GroundSurface* – půdorys budovy, *WallSurface* – stěny budovy, *RoofSurface* – střecha budovy. Následně byly za pomoci funkce *GMLPoints* (Biljecki 2015b) získány jednotlivé vertexy daných polygonů, ze kterých pak byly počítány další parametry budovy.

4.4.2 Výpočet potřebných parametrů budovy

Pro výpočet jednotlivých parametrů, byly vytvořeny funkce v programovacím jazyce *Python*. Níže budou popsány názvy jednotlivých funkcí a jejich funkcionalita.

Pro účely této práce byla každá budova pomyslně rozdělena na dvě části: tělo budovy a střešní část. Tělo je definováno jako ortogonální těleso, tedy objekt krychlového či

kvádrového typu. Je to půdorys budovy a na něj kolmé stěny. Střešní část je oddělena v místě, kdy se z ortogonálních ploch stávají plochy šikmé. Součet objemu těla budovy a střešní plochy udává vždy konečný objem celé budovy. Stejně tak je tomu i u výšky těla a střešní části budovy. Rozdělení budovy na tělo a střešní část u jednotlivých typů střech je uvedeno na obrázku č. 6. Všechny vzorce uvedené v této kapitole vychází z (Bartsch 1983).



Obrázek č. 6: Grafické znázornění rozdělení budovy,
tělo (šedá) a střešní část (červená)
(Zdroj: vlastní tvorba)

`parameters_of_footprint (footprintcoords)`

Ze souřadnic půdorysu vypočítá stranu *a, b* a následně spočítá obsah půdorysu:

$$S = a * b$$

Vrátí stranu *a, b*, dále *Z* souřadnici půdorysu (v modelech v této práci leží půdorys vždy v jedné rovině) a obsah půdorysu. Funkce počítá i s možností, že strany půdorysu nejsou rovnoběžné se souřadným systémem.

`Bodyvolume (wallcoords, footprintcoords)`

Vypočítá výšku těla budovy z listu souřadnic stěn kolmých na půdorys. Následně vypočítá objem těla budovy:

$$V = a * b * c$$

Vrátí objem těla budovy.

`Roofvolume (footprintcoords, rooftype, roofcoords)`

Vypočítá výšku pouze střešní části, odečtením maximální výšky těla budovy od půdorysu od max. výšky střechy budovy od půdorysu. Na základě typu střechy spočítá její objem:

Rovná (*flat*)

$$V = 0$$

Šikmá (*shed*) a **sedlová** (*gabled*) jedná se o polovinu krychle výšky střechy

$$V = \frac{1}{2}(a * b * c)$$

Stanová (*pyramidal*) objem jehlanu

$$V = \frac{1}{3}(S * v)$$

Valbová (*hipped*) objem klínu

$$V = \frac{1}{6} * b * v(2a + a1)$$

Vrátí objem střechy budovy.

`allvolume(roofvolume,bodyvolume)`

Vrátí sečtený objem střechy a „těla“ budovy.

`height_of_object(coords)`

Určí výšku objektu odečtením nejvyšší a nejnižší Z souřadnice a tuto hodnotu vrátí. Ve skriptu se používá pro určení výšky těla budovy a střešní části budovy, jejichž součtem vznikne výška celé budovy.

`neighbour_buildings(footprintcoords,anotherbuild)`

Na základě velikosti průniku bufferu půdorysů budov určí sousední budovy. Velikost bufferu a mezní hodnota obsahu průniku jsou volitelnými parametry. Vrátí ID sousedních budov.

`create_edges_from_list_of_connection(bld_nb)`

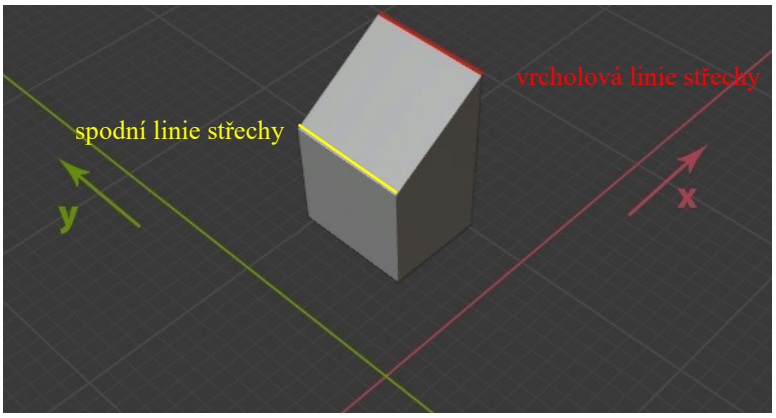
Z listu sousedství vytvořeného funkcí **`neighbour_buildings`** vytvoří list s vnořenými dvouprvkovými neměnnými seznamy, které označují spojení mezi sousedními budovami.

`roof_orientation(footprintcoords, rooftype, roofcoords)`

Na základě typu střechy určí orientaci vrcholové linie střechy. Orientace je určována vzhledem k ose *x* lokálního souřadného systému. Funkce vychází z předpokladu, že ve vstupním datovém modelu existují pouze dva typy natočení a všechny budovy se nacházejí v prvním kladném kvadrantu souřadné soustavy. Buď je střešní linie kolmá na osu *x*, nebo je s osou *x* rovnoběžná. Pro rovnou a stanovou střechu je orientace vůči ose x nastavena na 0°.

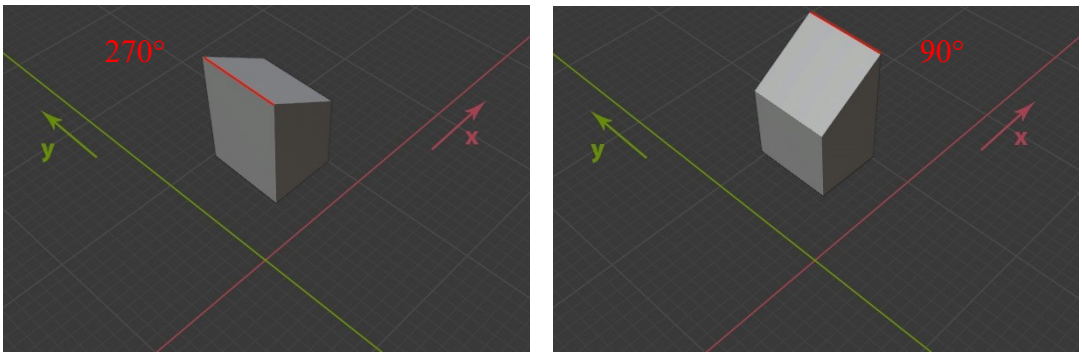
U valbové a sedlové střechy jsou odečteny koncové souřadnice vrcholové linie střechy. Pokud se: $|x_1 - x_2| = 0$, je budově nastavena orientace vůči ose *x* 90°. Pokud se: $|y_1 - y_2| = 0$, je budově nastaveno natočení vůči ose *x* 0°.

U pultových střech ještě záleží na orientaci vrcholové linie vůči spodní rovnoběžné linii střechy, viz obrázek č. 7.



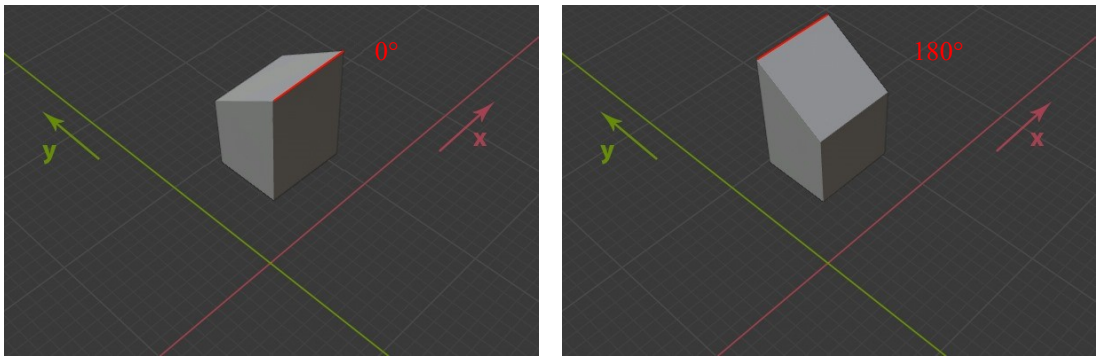
Obrázek č. 7: Definice důležitých hran při zjišťování orientace střechy (zdroj: vlastní tvorba)

Nejprve je zjištěno, jestli je vrcholová linie pultové střechy kolmá, nebo rovnoběžná s osou *x*, obdobně jak je tomu u valbové či sedlové střechy. Následně je na základě souřadnic koncových bodů spodní linie střechy zjišťováno, kde vrcholová linie leží. Pro budovy jejichž vrcholová linie je kolmá na osu *x*, je zjišťováno, zda je vrcholová linie blíže k ose *y* než spodní rovnoběžná linie. Pokud ano, je budově nastaveno 270°, v opačném případě je orientace nastavena na 90°, viz obrázek č. 8.



Obrázek č. 8: Určení orientace vrcholové linie střechy, pokud je vrcholová linie kolmá na osu x
(zdroj: vlastní tvorba)

Pro budovy jejichž vrcholová linie je rovnoběžná s osou *x* je zjišťováno, zda vrcholová linie leží blíže k ose *x* než spodní rovnoběžná linie. Pokud vrcholová linie leží blíže k ose *x*, je její orientace nastavena na 0°. V opačném případě je orientace nastavena na 180°, viz obrázek č. 9.



Obrázek č. 9: Určení orientace vrcholové linie střechy, pokud je vrcholová linie rovnoběžná s osou x
(zdroj: vlastní tvorba)

Skript vrátí orientaci střechy dané budovy.

Následně je vypočítána konstanta objemu střešní části, pomocí dvou vzniklých slovníků. Slovníky jsou vytvořeny ve formátu: klíč = ID budovy, proměnnou je počítaný parametr budovy. Vstupem jsou dva slovníky, jeden s vypočítanými objemy střešní části budov a druhý s vypočítanými výškami střešní části budov. Na základě ID budovy vydělí vypočítaný objem střešní části budovy s výškou střešní části budovy. Dále jsou upraveny některé parametry tak, aby mohly vstoupit do optimalizační funkce.

U jednotlivých budov jsou známy tyto parametry: ID budovy, typ střechy normalizovaný číslem (normalizace typu střešní plochy bude podrobně rozebrána v následujících kapitolách), obsah půdorysu, Z souřadnice půdorysu a Z souřadnice nejvyššího bodu těla a střešní části budovy. Dále je znám objem těla budovy a střechy budovy a jejich součet, výška budovy se střechou a bez ní, ID sousedních budov, orientace střešní části budovy a konstanta objemu střešní části budovy.

Výstupem skriptu jsou dva textové soubory. Jeden slouží jako vstup do optimalizační úlohy a má název *optimization_input.txt*. Druhý slouží jako vstup do skriptu, umožňující vizualizaci optimalizační úlohy s názvem: *z_visualization.txt*. Tyto soubory jsou uvedeny v digitální příloze této práce.

5. Lineární programování – optimalizační úloha – problém agregace

V následujících kapitolách bude popsáno, jakým způsobem přistupovat k problému agregace budov pomocí matematické optimalizace, přesněji lineárního programování. V kapitole *5.1* jsou popsány teoretická východiska pro agregaci bloků budov LOD1, vycházející z (Guercke a kol. 2011). V kapitole je definována objektivní funkce, ze které částečně vychází i optimalizační úloha vytvořená v této práci. Dále je zavedeno několik proměnných a nezbytných omezujících podmínek. V následující kapitole *5.2* je popsáno rozšíření, kdy jsou vstupními daty bloky budov LOD2, konkrétně zavedení několika omezujících podmínek a rozšíření objektivní funkce.

V kapitole *5.3* je popsána implementace problému v programovacím jazyce *Python* za použití nástrojů knihovny *Pulp*. V následující kapitole *5.4* jsou vysvětleny hodnoty volitelných proměnných optimalizační úlohy. Kapitola *5.5* pak poukazuje na možnosti rozšíření optimalizační úlohy.

5.1 Teoretická východiska optimalizační úlohy na bloky budov LOD1

Základní myšlenkou celé optimalizační úlohy je sloučit co nejvíce budov do agregačních celků. Dále pak minimalizovat objemové změny při agregaci jednotlivých budov do agregačních celků. To přináší celou řadu podmínek, které budou následně rozebrány.

Při agregaci budov je důležité rozhodnout, které budovy budou připojeny k sobě. Z důvodu definice, která budova náleží jakému celku, je nutné zavést binární proměnnou X_{uv}, která značí příslušnost budovy v k centru u. Množina B značí skupinu všech budov v datovém modelu. Základní myšlenkou je, že každá budova může být centrem a zároveň každé centrum může pojmout všechny budovy. To znamená, že existuje B^2 možností, které mohou nastat. Avšak za určitých podmínek.

Každá budova může být přiřazena pouze jednomu celku (C_b1, Guercke a kol. 2011):

$$\forall v \in B: \sum_{u \in B} X_{uv} = 1$$

Budova v může být přiřazena k centru u jedině pod podmínkou, že u je centrem (C_b2, Guercke a kol. 2011):

$$\forall v, u \in B: \; X_{uv} \leq X_{uu}$$

Pro vytvoření správné optimalizační úlohy je nutné znát sousednost jednotlivých budov, aby k sobě byly připojeny pouze budovy, které spolu sousedí. Budova může být přiřazena k jiné právě tehdy, pokud jsou budovy bezprostředně blízko sebe a mezi nimi není žádný další objekt. Pro popis sousednosti budov je nejvhodnější vytvořit graf, jehož uzly znázorňují jednotlivé budovy. Pokud mezi uzly existují hrany, znamená to, že spolu budovy sousedí, v opačném případě mezi uzly neexistuje propojení.

Pro sousední budovy *u, v* bude tedy existovat hrana *(u, v)* a *(v, u)*. Všechny tyto hrany jsou přidány do množiny *A*, jež značí všechny hrany v grafu sousednosti *N*. Každá budova, která není centrem generuje odtok hodnoty jedna, a po průchodu další hranou jsou hodnoty sčítány. Centrum přijímá veškerý přítok a negeneruje žádný pozitivní odtok. Pro definici optimalizačního omezení jsou definovány dvě proměnné F_a a f_a. f_a je spojitá proměnná značící sumu odtoku na dané hraně, je vždy větší nebo rovna 0. F_a je binární proměnná. Pokud na hraně existuje pozitivní tok $F_a=1$, v opačném případě $F_a=0$.

První podmínkou je, že F_a může být jedině 0 pokud je $f_a=0$ (C_f2, Guercke a kol. 2011):

$$\forall a \in A: \; MF_a \leq f_a$$

Kde *M* je volitelné číslo, větší než suma maximálního možného toku.

Rozdíl sumy odtoku a přítoku bude v uzlech, které nejsou centrem 1. V uzlech, které jsou centrem se budou hodnoty pohybovat v rozmezí -*M* až 0 (C_f2, Guercke a kol. 2011):

$$\forall v \in B: \sum_{a=(v,u)\in A} f_a - \sum_{a=(u,v)\in A} f_a \geq 1 - X_{vv}(M+1)$$

$$\sum_{a=(v,u)\in A} f_a - \sum_{a=(u,v)\in A} f_a \leq 1 - X_{vv}$$

Pokud na hraně existuje pozitivní tok, uzly na obou koncích hrany náleží stejnému centru (C_f3, Guercke a kol. 2011):

$$\forall (u, v) \in A, \forall c \in B: \; X_{cu} \geq X_{cv} + (F_{(u,v)} - 1)$$

Pouze jedna hrana zdroje produkuje pozitivní tok. Centrum pozitivní tok neprodukuje (C_f4, Guercke a kol. 2011):

$$\forall v \in B\colon X_{vv} + \sum_{a=(u,w)\in A} F_a \leq 1$$

Určení výsledné výšky agregovaných budov je jednou z klíčových otázek v optimalizační úloze. Nejprve je nutné definovat proměnné. H_u značí výšku výsledného agregátu s centrem u a je definována jako spojitá proměnná. Výšku původní budovy značí proměnná $h(v)$. Objemové změny značí spojitá proměnná ΔV_{uv}. Objemové změny a výška výsledného agregátu, jsou hodnoty, které jsou hledány pomocí optimalizační úlohy a jsou předem neznámé, spodní hranice intervalu těchto proměnných je nastavena na nulu. Objemové změny jsou počítány pouze v případě, pokud $X_{uv}=1$. V opačném případě $\Delta V_{uv}=0$ ($C_\Delta V$, Guercke a kol. 2011):

$$\forall v \in B\colon \Delta V_{uv} \geq (Bh(v) - H_u)A(v) - (1 - X_{uv})MB_{vol.v}$$

$$\Delta V_{uv} \geq -(Bh(v) - H_u)A(v) - (1 - X_{uv})MB_{vol.v}$$

Kde $M_{vol.\ v}$ je libovolné číslo, větší než je největší možná objemová změna, která může pro danou budovu v nastat.

Jak již bylo výše zmíněno, celá optimalizační úloha se snaží minimalizovat počet agregátů. Současně se také snaží minimalizovat objemové změny budov, které jsou připojeny k agregátu. Tato podmínka není tolik podstatná jako to, aby se minimalizoval počet agregátů. Z toho důvodu jsou objemové změny násobeny parametrem W, aby jejich výsledek byl menší než 1 a objemové změny neměly na výslednou agregaci takový vliv. Objektivní funkce budov LOD1 je definována jako (*objective_function*, Guercke a kol. 2011):

$$min\colon \sum_{v\in B} X_{vv} + W \sum_{u.v\in B} \Delta V_{uv}$$

5.2 Rozšíření optimalizačního problému na LOD2

Předchozí kapitola pojednává o tom, jakým způsobem je možné pohlížet na agregaci budov LOD1. V této kapitole bude uvedeno její rozšíření, kdy jsou vstupní data rozšířena

na LOD2. Největším rozdílem těchto úrovní je to, že budovy LOD1 jsou reprezentovány pouze ortogonálními tělesy různých tvarů – v této práci se jedná pouze o objekty typu krychle nebo kvádr. Zatímco budovy úrovně LOD2 jsou rozšířeny o typ střechy. S tím souvisí také změna objemu tělesa, orientace střechy, navíc budova již není ortogonální.

Hlavní myšlenkou rozšíření optimalizačního problému je přidání omezujících podmínek, týkajících se jednotlivých parametrů, které se vážou na střechy jako je: typ střechy, objem střechy, orientace, výška. Tento problém je potřeba rozdělit a popsat podrobněji.

Původním návrhem bylo, slučovat pouze střechy se stejným typem. Byla zavedena proměnná *Rt* označující typ střechy. Byla sestavena pravdivostní tabulka, za jakých podmínek je možné slučovat jednotlivé budovy, viz tabulka č. 1. To platí za podmínky, že *Rt>0.*

X_{uv}	*Rt(u) =Rt(v)*	*Validita*
0	0	1
0	1	1
1	0	0
1	1	1

Tabulka č. 1: Pravdivostní tabulka

Tabulka popisuje, že je nutné přidat omezení, které vylučuje situaci kdy $X_{uv} = 1$, ale zároveň střecha budovy, která je středem *R(u)* není shodná se střechou zkoumané budovy *R(v).* Z pravdivostní tabulky vychází, že jde o implikaci. Tedy:

$$X_{uv} \Rightarrow Rt(u) = Rt(v)$$

Pomocí De Morganových zákonů lze upravit (Bartsch 1983):

$$A \Rightarrow B \;\Leftrightarrow \bar{A} \vee B \Leftrightarrow \overline{A \wedge \bar{B}}$$

Z tabulky vyplývá, že jde o konjunkci, to je logický součin. Pro naše řešení:

$$X_{uv} \wedge Rt(u) \neq Rt(v)$$

$$X_{uv}\left(Rt(v) - Rt(u)\right) = 0 \qquad \forall x \epsilon \text{ B: } Rt(u), Rt(v) > 0$$

Podmínku lze následně rozšířit o hodnoty v intervalu epsilon. Tak aby nebyla příliš „tvrdá“ a neslučovala pouze stejné střechy. Pokud definujeme typ střechy jako lineární proměnnou, lze dále rozšířit omezení na (*rooftypes*):

$$X_{uv}\left(Rt(v) - Rt(u)\right) \leq \varepsilon \qquad \forall x \, \epsilon \text{ B: } Rt(u), Rt(v) > 0$$

$$X_{uv}\left(Rt(v) - Rt(u)\right) \geq -\varepsilon \qquad \forall x \, \epsilon \text{ B: } Rt(u), Rt(v) > 0$$

Pro které bude ε volitelný interval v námi omezeném rozmezí. Základním pravidlem je definovat zbylé parametry jako lineární proměnnou a určit vhodný interval ε.

Pro výše zmíněných pět typů střech tedy platí:

$$\begin{matrix} hodnota\ R \\ typ\ střechy \end{matrix} : \begin{matrix} 1 \\ stanová \end{matrix} \rightarrow \begin{matrix} 2 \\ rovná \end{matrix} \rightarrow \begin{matrix} 3 \\ pultová \end{matrix} \rightarrow \begin{matrix} 4 \\ sedlová \end{matrix} \rightarrow \begin{matrix} 5 \\ valbová \end{matrix}$$

Stanová střecha může přejít zmenšováním výšky vrcholového bodu na rovnou. Z rovné se při kolmém zvednutí jedné boční hrany může stát pultová. Pokud se vrcholová hrana pultové střechy přesune na střed těla budovy, jde o střechu sedlovou. A pokud dojde ke zmenšení vrcholové hrany sedlové střechy, jde o valbovou střechu.

Následně lze obdobně do omezujících podmínek přidat další parametry jako je orientace střechy a agregovat pouze střechy s podobnou orientací. Proměnná *Ro* značí orientaci budovy (*rooforientation*):

$$X_{uv}\left(Ro(v) - Ro(u)\right) \leq \varepsilon_{Ro} \qquad \forall x \, \epsilon \text{ B: } Ro(u), Ro(v) > 0$$

$$X_{uv}\left(Ro(v) - Ro(u)\right) \geq -\varepsilon_{Ro} \qquad \forall x \, \epsilon \text{ B: } Ro(u), Ro(v) > 0$$

Proměnná ε_{Ro} značí maximální úhel natočení vrcholové střešní linie vůči ose *x* v lokálním souřadném systému, pro který je možné ještě agregovat budovy. Jak již bylo zmíněno, v datových modelech vytvořených pro tuto práci existují pouze dvě varianty natočení. Buď je vrcholová střešní linie natočena kolmo na osu *x* nebo je s osou *x* rovnoběžná. Střechy, u kterých nezáleží na jejich orientaci, tedy rovná a stanová, mají automaticky nastavenou orientaci na 0°.

Dále jsou obdobně zavedeny další podmínky. Je možné agregovat pouze takové budovy, jejichž rozdíl výšek nebude přesahovat nastavený parametr. Tak je tomu zvlášť

pro výšku těla budovy *Bh* (*hard_body_height*) a pro výšku střechy budovy *Rh* (*hard_roof_height*):

$$X_{uv}\left(Bh(v) - Bh(u)\right) \leq \varepsilon_{Ro} \qquad \forall x \in \mathrm{B}: Bh(u), Bh(v) > 0$$

$$X_{uv}\left(Bh(v) - Bh(u)\right) \geq -\varepsilon_{Ro} \qquad \forall x \in \mathrm{B}: Bh(u), Bh(v) > 0$$

$$X_{uv}\left(Rh(v) - Rh(u)\right) \leq \varepsilon_{Ro} \qquad \forall x \in \mathrm{B}: Rh(u), Rh(v) > 0$$

$$X_{uv}\left(Rh(v) - Rh(u)\right) \geq -\varepsilon_{Ro} \qquad \forall x \in \mathrm{B}: Rh(u), Rh(v) > 0$$

Jak již bylo uvedeno v kapitole *4.4.2*, budova je pomyslně rozdělena na dvě části. Spodní část, která je označována jako tělo budovy a zjednodušeně jde o úroveň detailu LOD1, vrchní část charakterizuje střechu daného objektu. Jde o těleso, kterému lze vypočítat výšku či objem. Z tohoto důvodu lze také jednoduše oddělit tyto dvě části v optimalizační úloze. To je dobré zejména proto, aby bylo z výsledků patrné, jakou výšku nastavit tělu výsledného agregátu a jakou výšku nastavit střešní části. V objektivní funkci lze pak při hledání minim objemovým změnám těla budovy nastavit jiný parametr, než minimalizaci objemových změn střech a určit tak jejich významnost v celé optimalizační úloze.

Nastavení omezujících podmínek pro objemové změny střešního objektu vycházejí z podmínky *CΔV* pro LOD1 (*ΔVroof*):

$$\forall v \in B:\ \Delta VR_{uv} \geq (Rh(v) - HR_u)K(v) - (1 - X_{uv})MR_{vol.v}$$

$$\Delta VR_{uv} \geq -(Rh(v) - HR_u)K(v) - (1 - X_{uv})MR_{vol.v}$$

Kde *K* je konstanta a je vypočítána pro každou budovu jako:

$$K(v) = \frac{V_R(v)}{Rh(v)}$$

Objemové změny střešních ploch rozšiřují objektivní funkci(*objective_function*):

$$min: \sum_{v \in B} X_{vv} + WB \sum_{u.v \in B} \Delta V_{uv} + WR \sum_{u.v \in B} \Delta VR_{uv}$$

5.3 Implementace optimalizační úlohy

Omezující podmínky a definice objektivní funkce, které jsou uvedeny v kapitole *5.1* a *5.2* byly implementovány pomocí optimalizační knihovny *Pulp* v jazyce *Python*, verzi *3.7.1*. Celý skript je dostupný na adrese: https://github.com/mechurk/optimization.

Nejprve je třeba definovat název celé optimalizační úlohy a také to, zda bude úloha hledat minima či maxima objektivní funkce. Tato úloha řeší minimalizaci agregátů za minimálních objemových změn, z toho vyplývá že jde o minimalizaci:

```
Lp_prob = p.LpProblem('Problem', p.LpMinimize)
```

Kód č. 1:Nastavení optimalizační úlohy

Skript je navržený tak, že přijímá předem známé parametry budov jako je: obsah podstavy, výška půdorysu a další, ve formě slovníků (*dictionary*). Pro každý parametr je definován jeden slovník. Klíčem slovníků je ID budov a proměnnou může být například výška nebo objem, který se váže ke konkrétnímu ID budovy. Takto jsou uloženy všechny potřebné parametry, které vstupují do optimalizační úlohy. Výjimku tvoří parametr sousednosti budov. Je to graf orientovaných hran mezi budovami. Ten je tvořen měnitelným seznamem (*list*), ve kterém je vnořený neměnný seznam (*tuple*), jenž obsahuje pouze dvě hodnoty. Ty reprezentují hranu grafu sousednosti jednotlivých budov. Pokud spolu dvě budovy nesousedí, pak mezi nimi neexistuje hrana. Další výjimkou je množina ID všech budov nacházejících se v datovém modelu, ta je tvořena měnitelným seznamem (*list*). Jednotlivé parametry vstupující do optimalizační úlohy včetně jejich datových typů jsou popsány v tabulce č. 2.

Skript přijímá vstupní parametry ve formě textového dokumentu pod názvem optimization_input.txt, který je výstupem skriptu výpočtu parametrů budovy. Skript výpočtu parametrů budovy je rozebrán v kapitole *4.4*.

Parametr	Název datového typu ve skriptu	Datový typ	Formát
ID budovy	building_ids	list	[ID1, ID2…IDn]
ID centra agregátů	center_ids	list	[ID1, ID2…IDn]
Výška těla budovy (Bh)	heights	dictionary	{ID1:h1,ID2:h2…IDn:hn}
Obsah půdorysu budovy (A)	footprints	dictionary	{ID1:A1,ID2:A2…IDn:An}
Hrana sousedsví (E)	edges	tuple in dictionary	[(ID1,ID2),(ID2,ID1)…(IDn,IDn)]
Typ střechy budovy (Rt)	roof_types	dictionary	{ID1:R1,ID2:R2…IDn:Rn}
Výška střechy budovy (Rh)	roof_heights	dictionary	{ID1:hR1,ID2:hR2…IDn:hRn}
Konstanta objemu střechy (K)	roof_volume _constant	dictionary	{ID1:K1,ID2:K2…IDn:Kn}
Orientace vrcholové hrany střechy (Ro)	roof_orientation	dictionary	{ID1:Ro1,ID2:Ro2…IDn:Ron}

Tabulka č. 2: výpis parametrů a jejich datových typů vstupujících do optimalizace

Dále jsou definovány proměnné, jejichž hodnoty nejsou předem známé. U každé proměnné je potřeba určit její název, nastavit spodní a horní hranici intervalu, ve kterém se výsledná hodnota bude pohybovat. Pokud nebudou tyto hodnoty nastaveny, je spodní hranice automaticky nastavena na $-\infty$ a horní hranice na $+\infty$. Dále se určuje, zda by mělo jít o celé číslo, binární či spojitou proměnnou. Pokud není nastaveno jinak, defaultně se nastavuje spojitá proměnná. V tabulce č. 3 jsou uvedeny všechny neznámé proměnné vstupující do optimalizační úlohy. Popis jednotlivých proměnných je uveden v kapitolách *5.1* a *5.2*.

Proměnná	Popis	Spodní hranice intervalu	Vrchní hranice intervalu	Kategorie
Xuv	náležitost budovy *v* k centru *u*	0	1	binární
ΔV	objemová změna těla budovy	0	+∞	spojitá
Hu	výška těla výsledného agregátu	0	+∞	spojitá
fa	suma pozitivního toku sousedství	0	+∞	spojitá
Fa	ukazatel pozitivního toku na hraně	0	1	binární
ΔVR	objemová změna střešní části budovy	0	+∞	spojitá
HRu	Výška střešní části výsledného agregátu	0	+∞	spojitá

Tabulka č. 3: Soupis neznámých proměnných vstupujících do optimalizační úlohy

Do optimalizační úlohy vstupují proměnné, které jsou vypočítány předem v optimalizačním skriptu a úzce souvisí s definovanými podmínkami, těmi jsou proměnné: M (použité v omezující podmínce *Cf2*), $MB_{vol.v}$ (použité v omezující podmínce $C_{\Delta}V$) a $MR_{vol.v}$ (použité v omezující podmínce $_{\Delta}Vroof$). Proměnná M je vypočítána jako počet budov v daném datovém modelu. $MB_{vol.v}$ a $MR_{vol.v}$ značí maximální možnou objemovou změnu, která může v daném bloku budov nastat. Jsou počítány zvlášť pro každou budovu.

Pro výpočet parametrů $M_{vol.v}$ je vytvořena funkce *calculate_M_vo_volume*. Vstupními parametry jsou hrany sousedství (*E)* a výška. Pokud jde o tělo budovy jedná se o výšku budovy (*h*), pokud jde o střešní část, jedná se o výšku střechy (*hR*). Dále do funkce vstupuje parametr, díky kterému je možné vypočítat výsledný objem tělesa. U těla budovy je to obsah půdorysu (*A)*. U střešní části je to konstanta (*K*), jejíž výpočet je popsán v kapitole *4.4.2*. Parametry $M_{vol.v}$ jsou vypočítány následujícím způsobem: nejprve je vytvořen neorientovaný graf z hran sousedství. Uzly grafu jsou ID budov, hrany značí sousednost mezi jednotlivými budovami. Následně jsou vypočítány jednotlivé komponenty grafu. Jedna komponenta charakterizuje blok budov. Pro každou komponentu je zvlášť odečtena výška nejvyšší budovy v bloku, nejnižší bod bloku je nastaven na 0. Následně je pro každou budovu v bloku vypočten rozdíl mezi výškou budovy a maximální výškou v bloku a rozdíl mezi výškou budovy a minimální výškou v bloku. Z těchto dvou hodnot je vybrána ta větší. Poté je hodnota násobena buď obsahem půdorysu (pro tělo budovy) nebo konstantou (pro střešní část budovy), tak aby výsledkem byla maximální možná objemová změna pro danou budovu. Funkce vrací proměnnou ve formě slovníku. Klíčem je ID dané budovy, proměnou maximální objemová změna:

```
def calculate_M_vo_volume(bld_nb, height, footprints):
    G = nx.Graph()
    for i in bld_nb:
        G.add_node(i[0])
        bld = i[0]
        for a in i:
            if a != bld and G.has_edge(bld, a) == 0:
                # print (a)
                G.add_edge(bld, a)

    n = nx.number_connected_components(G)
    con = nx.connected_components(G)
    cc = list(con)

    M = {}
    M_all = {}
    for block in cc:
        heights_block = [0]
        for building in block:
            for h in height:
                if h == building:
                    heights_block.append(height[h])
        max_block = max(heights_block)
        min_block = min(heights_block)

        for building in block:
            h_set = []
            for h in height:
                if h == building:
                    minimum = height[h] - min_block
                    maximum = max_block - height[h]
                    h_set.append(minimum)
                    h_set.append(maximum)
                    total_h = max(h_set)
                    M[h] = total_h
    for par in M:
        for footprint in footprints:
            if par == footprint:
                M_final = M[par] * footprints[footprint]
                M_all[footprint] = M_final

    return (M_all)
```

Kód č. 2: Výpočet parametru $M_{vol.v}$

Dalšími důležitými proměnnými jsou volitelné parametry, ovlivňující výsledek optimalizační úlohy. Jde o všechny proměnné ε objevující se v omezujících podmínkách a o proměnné *WB a WR* vstupující do objektivní funkce. Tyto proměnné jsou nastavovány řešitelem optimalizační úlohy tak, aby výsledek optimalizační funkce vedl k požadovaným výsledkům. Blíže budou rozebrány v kapitole *5.4*.

Každá omezující podmínka z kapitol *5.1* a *5.2* je definována samostatnou funkcí, která má stejný název jako daná podmínka. Pomocí funkce je omezující podmínka přidána do optimalizační úlohy:

```
def cb1_one_building_id_all_center_ids(Lp_prob, building_id, center_ids):
    Lp_prob += p.lpSum(center_matrix[center_id, building_id] for center_id in
center_ids) == 1


def cb1(Lp_prob, building_ids, center_ids):
    for building_id in building_ids:
        cb1_one_building_id_all_center_ids(Lp_prob, building_id, center_ids)


def cb2_two_building_ids_one_center_id(Lp_prob, first_building_id,
second_building_id, center_id):
    Lp_prob += center_matrix[center_id, first_building_id] <=
center_matrix[center_id, second_building_id]


def cb2(Lp_prob, building_ids, center_ids):
    for index in (range(len(building_ids))):
        for first_building_id in building_ids:
            cb2_two_building_ids_one_center_id(Lp_prob, first_building_id,
building_ids[index], center_ids[index])


def cf1(Lp_prob, edges, flows, positive_flows):
    for edge in edges:
        Lp_prob += building_count * positive_flows[edge] >= flows[edge]


def cf2(Lp_prob, flows, building_ids):
    for building_id in building_ids:
        outcoming_edges = [edge for edge in edges if edge[0] == building_id]
        incoming_edges = [edge for edge in edges if edge[1] == building_id]
        Lp_prob += p.lpSum(flows[edge] for edge in outcoming_edges) - p.lpSum(
            flows[edge] for edge in incoming_edges) >= 1-
center_matrix[building_id, building_id] * (
                            building_count + 1)
        Lp_prob += p.lpSum(flows[edge] for edge in outcoming_edges) - p.lpSum(
            flows[edge] for edge in incoming_edges) <= 1 -
center_matrix[building_id, building_id]


def cf3(Lp_prob, edges, building_ids, positive_flows):
    for edge in edges:
        for building_id in building_ids:
            Lp_prob += center_matrix[building_id, edge[0]] >=
center_matrix[building_id, edge[1]] + (
                    positive_flows[edge] - 1)


def cf4(Lp_prob, building_ids, edges, positive_flows):
    for building_id in building_ids:
        outcoming_edges = [edge for edge in edges if edge[0] == building_id]
        Lp_prob += center_matrix[building_id, building_id] + p.lpSum(
            positive_flows[edge] for edge in outcoming_edges) <= 1
```

```
def c_delta_V_one_building_one_center(Lp_prob, building_id, center_id):
    Lp_prob += delta_volumes_matrix[center_id, building_id] >=
(heights[building_id] - height_center[center_id]) * \
               footprints[building_id] - (
                       1 - center_matrix[center_id, building_id]) *
MB[building_id]
    Lp_prob += delta_volumes_matrix[center_id, building_id] >= -
(heights[building_id] - height_center[center_id]) * \
               footprints[building_id] - (
                       1 - center_matrix[center_id, building_id]) *
MB[building_id]


def c_delta_V(Lp_prob, building_ids, center_ids):
    for building_id in building_ids:
        for center_id in center_ids:
            c_delta_V_one_building_one_center(Lp_prob, building_id,
center_id)
```

Kód č. 3 Implementace omezujících podmínek první část

Obdobně tomu je i u rozšiřujících omezujících podmínek pro budovy LOD2:

```
def rooftypes(Lp_prob, center_ids, buiding_ids, roof_types):
    for center_id in center_ids:
        for building_id in building_ids:
            Lp_prob += center_matrix[center_id, building_id] * (
                    roof_types[center_id] - roof_types[building_id]) <=
epsilon_roof_type
            Lp_prob += center_matrix[center_id, building_id] * (
                    roof_types[center_id] - roof_types[building_id]) >= -
epsilon_roof_type

def hard_body_height(Lp_prob, center_ids, buiding_ids, heights):
    for center_id in center_ids:
        for building_id in building_ids:
            Lp_prob += center_matrix[center_id, building_id] * (
                    heights[center_id] - heights[building_id]) <=
epsilon_height
            Lp_prob += center_matrix[center_id, building_id] * (
                    heights[center_id] - heights[building_id]) >= -
epsilon_height

def hard_roof_height(Lp_prob, center_ids, buiding_ids, roof_heights):
    for center_id in center_ids:
        for building_id in building_ids:
            Lp_prob += center_matrix[center_id, building_id] * (
                    roof_heights[center_id] - roof_heights[building_id]) <=
epsilon_roof_height
            Lp_prob += center_matrix[center_id, building_id] * (
                    roof_heights[center_id] - roof_heights[building_id]) >=
-epsilon_roof_height
```

```
def delta_v_roof_one_building_one_center(Lp_prob, building_id, center_id):
    Lp_prob += delta_roofs_volume_matrix[center_id, building_id] >= (
            roof_heights[building_id] - roofs_height_center[center_id]) *
roof_volume_constant[building_id] - (
                        1 - center_matrix[center_id, building_id]) *
MR[building_id]
    Lp_prob += delta_roofs_volume_matrix[center_id, building_id] >= -(
            roof_heights[building_id] - roofs_height_center[center_id]) *
roof_volume_constant[building_id] - (
                        1 - center_matrix[center_id, building_id]) *
MR[building_id]


def delta_v_roof(Lp_prob, building_ids, center_ids):
    for building_id in building_ids:
        for center_id in center_ids:
            delta_v_roof_one_building_one_center(Lp_prob, building_id,
center_id)


def rooforientation(Lp_prob, center_ids, buiding_ids, roof_orientation):
    for center_id in center_ids:
        for building_id in building_ids:
            Lp_prob += center_matrix[center_id, building_id] * (
                    roof_orientation[center_id] -
roof_orientation[building_id]) <= epsilon_roof_orientation
            Lp_prob += center_matrix[center_id, building_id] * (
                    roof_orientation[center_id] -
roof_orientation[building_id]) >= -epsilon_roof_orientation
```

Kód č. 4: Implementace omezujících podmínek druhá část

Následně je definována objektivní funkce celé optimalizační úlohy a je přidána do výpočtu:

```
def objective_function(Lp_prob, building_ids, center_ids):
    Lp_prob += p.lpSum(
        center_matrix[center_ids[index], building_ids[index]] for index in
        range(len(building_ids))) + (body_volume_change_weight * p.lpSum(
        delta_volumes_matrix)) + (roof_volume_change_weight * p.lpSum(
        delta_roofs_volume_matrix))
```

Kód č. 5: definice objektivní funkce

Pro zobrazení výsledků původně neznámých proměnných je zavedena funkce *printProb*:

```
def printProb(Lp_prob):
    for v in Lp_prob.variables():
        print(v.name, "=", v.varValue)
    print("Status:", p.LpStatus[Lp_prob.status])
```

Kód č. 6: výpis původně neznámých proměnných

Výsledkem optimalizačního skriptu je výpis všech původně neznámých proměnných s hodnotou výsledku optimalizační úlohy. Dále je vypsán výsledek objektivní funkce a status dané úlohy. Existuje pět typů statusu kódu:

- *Not Solved* – status před řešením problému
- *Optimal* – bylo nalezeno optimální řešení – žádoucí stav
- *Infesiable* – neexistuje žádné řešení
- *Unbounded* – existuje nekonečně mnoho řešení
- *Undefined* – může existovat optimální řešení, ale nemusí být nalezeno

Status slouží ve skriptu jako kontrola, že bylo nalezeno optimální řešení. Daný skript totiž pokaždé vypočítá hodnoty neznámých proměnných.

Hodnoty všech původně neznámých proměnných jsou po správném vyřešení optimalizační úlohy vyexportovány do textového souboru s názvem *„optimization_solve.txt“*. Výsledek optimalizační úlohy mimo jiné obsahuje matici čísel, udávající, které budovy mají být agregovány, jaká je výška výsledného agregátu, jaké jsou objemové změny budovy atd.

Knihovna Pulp částečně upravuje hodnoty proměnných a je nutné s touto skutečností počítat, například pro další práci s daty. Tou může být vizualizace výsledků práce. Vstupními daty v této práci jsou ID budov, podobného formátu např.:

ID = 0e09ee42- 5d0a- 4787- 8e33- 4c1763561d3c.

Vždy jde o sadu písmen a čísel oddělených pomlčkou. Tyto pomlčky byly po výpočtu optimalizační úlohy automaticky nahrazeny podtržítky.

5.4 Hodnoty volitelných proměnných optimalizační úlohy

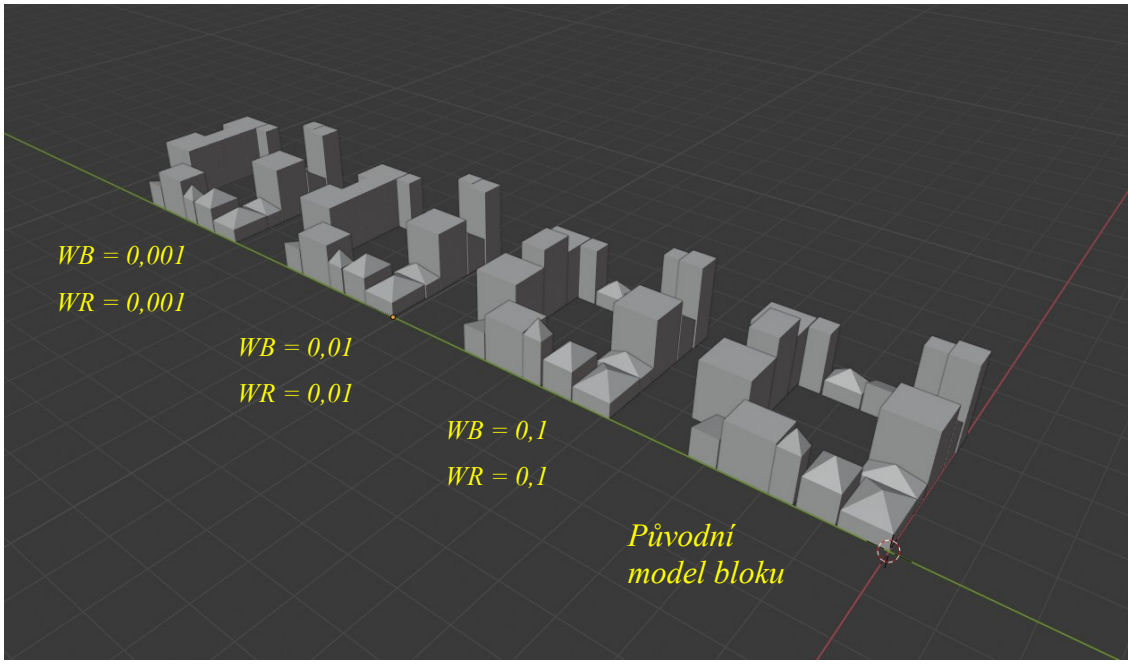
Ve vytvořené optimalizační úloze existují dva typy volitelných proměnných. Jedním typem jsou proměnné *WB a WR* vstupující do objektivní funkce. Druhým typem jsou všechny proměnné *epsilon* vstupující do omezujících podmínek.

Proměnné *epsilon* udávají, jaké budovy se můžou ještě agregovat do jednoho celku a jaké už ne, na základě podobných charakteristik budov. Obecně jde o princip, kdy jsou základní charakteristiky definovány jako spojité proměnné. A hodnota epsilon udává mezní rozdíl charakteristik dvou budov, aby mohly být ještě agregovány. Níže budou rozepsány jednotlivé proměnné epsilon a také charakteristiky budovy, které k nim náleží.

- *Epsilon_roof_type* – modely budov v této diplomové práci obsahují pět typů střech, viz kapitola *4.2*. Typy střech byly následně normalizovány na celá čísla v intervalu $\langle 1{,}5 \rangle$, viz kapitola *5.2*. Hodnota epsilon by měla tedy nabývat celých čísel v intervalu $\langle 0{,}5 \rangle$. Pokud bude hodnota epsilon 5, v optimalizační úloze nebude záležet na typu střechy. Pokud bude hodnota epsilon 0, mohou se agregovat pouze budovy se stejným typem střechy.
- *Epsilon_roof_orientation* – v tomto případě uvádí epsilon rozdíl natočení ve stupních vrcholové střešní linie vůči lokálnímu souřadnému systému, kdy mohou být budovy ještě agregovány. Obecně může nabývat proměnná natočení vrcholové střešní linie hodnot v intervalu $\langle 0°, 359° \rangle$ a hodnota epsilon pak hodnot v intervalu $\langle 0{,}359 \rangle$. V modelech vytvořených pro tuto práci, mohou vrcholové střešní linie nabývat pouze hodnot 0° a 90° pro valbové a sedlové typy střech. Pro pultový typ střechy 0°, 90°, 180° a 270°. U rovné a stanové střechy na orientaci nezáleží a hodnoty orientace jsou nastaveny na 0°. Při zjišťování orientace vrcholové střešní linie, je nutné brát v potaz typ střechy a u sedlových a valbových střech sjednotit identické orientace opačného směru (např. 0° = 90°).
- *Epsilon_height* – v tomto případě hodnota epsilon uvádí maximální rozdíl výšek těla budovy, kdy mohou být budovy ještě agregovány. Interval epsilon závisí na minimální a maximální výšce budovy v modelu.
- *Epsilon_roof_height* – epsilon funguje na stejném principu jako u *Epsilon_height*, s tím rozdílem, že proměnnou je výška střešní části budovy nikoliv těla budovy.

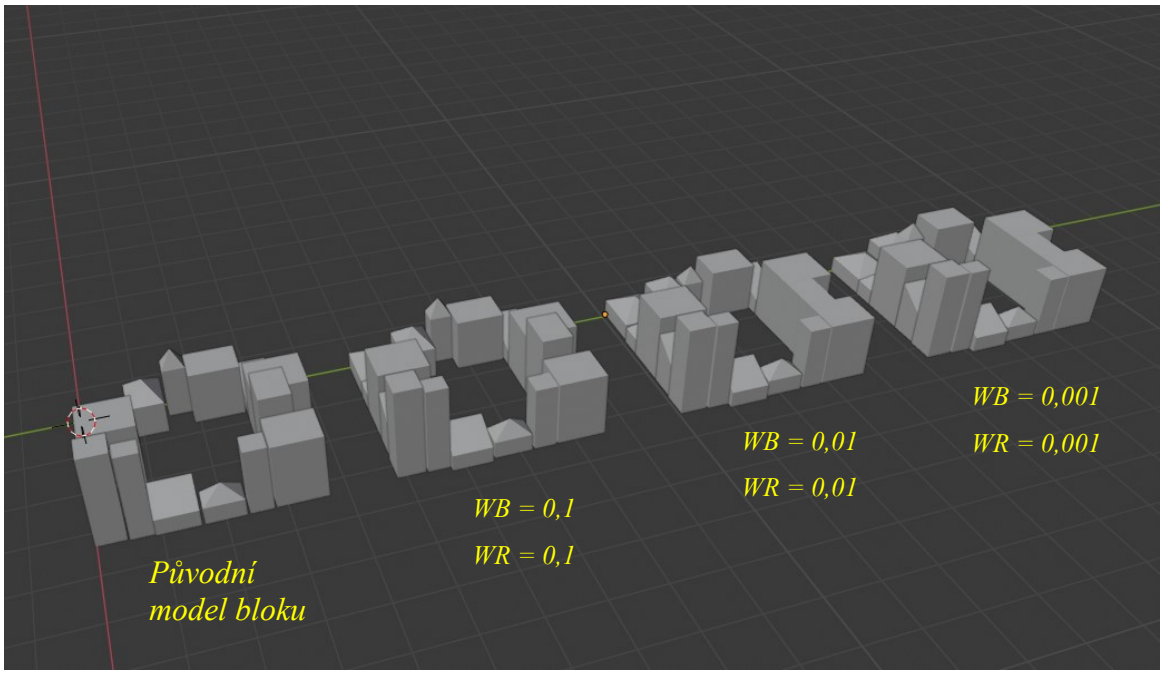
Zatímco u proměnných epsilon je poměrně jednoduché odvodit, jaký vliv bude mít jejich hodnota na výsledek optimalizační úlohy, u proměnných vstupujících do objektivní funkce je to složitější. Jde o hodnoty určující, jaký vliv bude mít objemová změna na výsledek celé optimalizační úlohy. *WB* je parametr určující vliv objemové změny těla budovy a *WR* je parametr určující vliv objemové změny střešní části. Jak již bylo zmíněno v předchozích kapitolách, objektivní funkce je nastavena tak, že se primárně snaží minimalizovat počet agregátů v daném modelu. Hodnoty proměnných *WB a WR* se tak nachází v intervalu (0,1). Přičemž čím se hodnota více blíží jedné, tím větší vliv mají objemové změny na výsledek celé optimalizační úlohy. Obrázky č. 10 a 11 ukazují, jak se mění výsledný model bloku v závislosti na změně *WB a WR*.

Z obrázků je patrné, že čím je menší parametr *W*, tím dojde k většímu „zarovnání“ budov i s velmi rozdílnými objemy. Naopak čím více se parametr *W* blíží jedné, model zdůrazňuje rozdílnost objemů v bloku.



Obrázek č. 10: Změna výsledku optimalizační funkce, při změně WB a WR, pohled z jedné strany

(Zdroj: vlastní tvorba)



Obrázek č. 11:Změna výsledku optimalizační funkce, při změně WB a WR, pohled z opačné strany

(Zdroj: vlastní tvorba)

5.5 Možné rozšíření optimalizační úlohy

Omezující podmínky umožňující agregovat pouze budovy, které mají podobný typ střechy, objem atd (viz kapitola *5.2*), lze sloučit do jedné tím, že dané proměnné převedeme na takové hodnoty, aby měly stejnou váhu (aby byly ve stejném intervalu hodnot např. 1 až 5). Následně lze omezení upravit podle toho, zda chceme, aby byly splněny všechny parametry, nebo stačí pouze jeden. Pro rozšíření o objem střechy (a další parametry) by platilo:

Logický operátor OR (+)

$$X_{uv}\left(\left(Rt(v)-Rt(u)\right)+\left(V_R(v)-V_R(u)\right)\ldots.\right)\leq\varepsilon$$

$$X_{uv}\left(\left(Rt(v)-Rt(u)\right)+\left(V_R(v)-V_R(u)\right)\ldots.\right)\geq-\varepsilon$$

Logický operátor AND (*)

$$X_{uv}\left(\left(Rt(v)-Rt(u)\right)*\left(V_R(v)-V_R(u)\right)\ldots.\right)\leq\varepsilon$$

$$X_{uv}\left(\left(Rt(v)-Rt(u)\right)*\left(V_R(v)-V_R(u)\right)\ldots.\right)\geq-\varepsilon.$$

Pak už jen stačí vhodně nastavit hodnotu ε. Při takto nastaveném omezení lze kombinovat logický operátor AND s logickým operátorem OR v jednom řádku. Šlo by tedy například

zadat podmínku: sloučit podobné typy střech které mají buď podobný objem nebo podobnou výšku.

Otázkou zůstává, zda je vhodné a vůbec možné převádět a následně sčítat takovou proměnnou jako je objem střechy s natočením či výškou střechy. Zda by výsledek optimalizační úlohy nebyl až příliš obecný.

Jedním z dalších možných rozšíření optimalizační úlohy může být výpočet sklonu střešních ploch a následná agregace budov pouze s podobným sklonem. Sklon střešní plochy je jedním z rozhodujících parametrů pro instalaci solárních panelů na střešní ploše. V případě určení solárního potenciálu střech je zase nutné znát co nejpřesnější obsah střešní plochy. V takovém případě je dobré minimalizovat plošné změny střechy. Většina definovaných střech v této práci není složena pouze z jedné střešní plochy. Je tedy nutné minimalizovat více ploch, a to ve stejném poměru. Dalším problémem je, že střešní plochy nejsou tvořeny stejnými geometrickými tělesy. Definice takové optimalizační úlohy je tedy značně netriviální.

Dále by bylo možné řešit vzájemnou orientaci střech vůči sobě, tedy ne pouze vůči lokálnímu souřadnému systému. Například u dvou pultových střech existuje šestnáct možností vzájemného natočení a pokud k tomu přidáme ještě rovnou střechu, existují možnosti, kdy jsou střechy orientované tak, že lze pultovou střechu sloučit s rovnou.

Další z možností rozšíření problému agregace budov je neřešit pouze výškovou změnu budov ale zaměřit se také na změnu půdorysu. To platí zejména u těla budovy. Výsledkem optimalizační úlohy je objemová změna dané budovy. Lze tedy dopočítat objemovou změnu celého agregátu a následně nový objem agregátu. Poté by bylo možné upravit půdorys a následně výšku těla budovy tak, aby se z půdorysu stal souvislý zjednodušený tvar, například obdélník.

Guercke a kol. (2011) ve svém článku rozšiřují optimalizační úlohu ještě o řadu omezujících podmínek, které by měly vést k lepšímu vizuálnímu vjemu agregovaného modelu budov. Jednou z takových podmínek je i zavedení několika omezení, aby výsledná výška agregátu nebyla menší, než je nejmenší výška vstupní agregované budovy. Tyto omezující podmínky obsahují dvě neznámé proměnné. Knihovna *Pulp*, ve které je daná optimalizační úloha vytvořena nepodporuje matematický operátor násobení mezi dvěma neznámými proměnnými. Z tohoto důvodu není podmínka v optimalizační úloze uvedena.

6. Vizualizace

Výsledkem optimalizační úlohy jsou číselné hodnoty. Není jednoduché představit si, jak tyto hodnoty změní výsledný model budov, a proto je dobré, výsledek optimalizační úlohy vizualizovat.

Z důvodu náročnosti modelování vstupních dat, získávání parametrů budov a definování optimalizační úlohy, již nebylo v silách této práce provézt podrobnou vizualizaci, spojenou s agregací jednotlivých ploch, vyhlazením půdorysu a stěn kolmých na půdorys. Navíc všechny zmíněné operace přímo nesouvisí s výsledky optimalizační úlohy. Jde o operace, které na optimalizační úlohu navazují. Podrobná vizualizace je samostatné téma, které lze oddělit od výsledků optimalizační úlohy. Z uvedených důvodů byly vizualizovány pouze změny výšek těla budovy a střešní části výsledných agregátů v daném bloku. Nejde o ideální řešení, avšak úplná vizualizace výsledku optimalizační úlohy přináší celou řadu otázek, které je nutné podrobněji prozkoumat.

Skript, umožňující vizualizaci optimalizační úlohy, je uveden na adrese https://github.com/mechurk/visualization_of_optimization. Skript je napsaný v programovacím jazyce *Python* ve verzi *3.7.1*. Požaduje na vstupu výsledné hodnoty optimalizační úlohy ve formě slovníku. Dále pak původní data, konkrétně *Z* souřadnici výšky těla budovy, *Z* souřadnici maximální výšky střešní plochy a *Z* souřadnici půdorysu budovy. Jak již bylo zmíněno výše, datové modely vytvořené pro tuto práci, se nachází, až na malé odchylky ve stejné rovině a půdorysy jednotlivých budov nejsou nijak nakloněny. Z toho vyplývá, že *Z* souřadnice maximálních výšek těla budovy i střešní části budou jednotné pro všechny plochy, které tvoří budovu. Dále je požadován původní datový model v *OBJ* formátu. Tento formát byl vybrán z důvodu jeho jednoduché struktury a také z důvodu, že ho podporuje značné množství softwarů a není tedy problém s jeho vizualizací, narozdíl od *GML* formátu.

Skript lze rozdělit na dvě části. První část extrahuje potřebná data z výsledku optimalizační úlohy a upravuje je tak, aby je bylo možné použít jako klíč plus nové hodnoty ve formě slovníku. Tato data jsou potřebná k zápisu do *OBJ* formátu. Výsledkem optimalizační úlohy je totiž slovník, kde je klíčem hodnota původně neznámé proměnné a proměnnou je výsledek optimalizační úlohy.

V první části jsou vyextrahovány hodnoty centrální matice – to je binární proměnná určující náležitost dané budovy k centru agregace. Obecně lze říci, že jde o proměnnou,

která určuje, jaké budovy se mají agregovat. Dále jsou extrahovány hodnoty výšky výsledného agregátu těla budovy a hodnoty výšky výsledného agregátu střešní části budovy. Zbytek první části lze definovat dvěma, respektive čtyřmi funkcemi (jsou to podobné funkce, jednou pro tělo budovy a podruhé pro střešní část budovy):

`create_dict_new_body_height (center_matrix, height_center)`

Ze slovníku nově vypočítaných center výšek těla agregátů (*height_center*) nalezne nenulové hodnoty. Parsuje hodnotu klíče slovníku center výšek těla agregátů tak, aby bylo odděleno ID budovy. Pokud se hodnota klíče nenulové proměnné nachází i v klíči slovníku, kde jsou uvedeny příslušnosti daných budov k centru agregátů(*center_matrix*) a proměnná u daného klíče je nenulová, klíč centrální matice je parsován tak, aby bylo vybráno pouze ID budovy, která má být agregována.

Následně je vytvořen nový slovník, jehož klíčem je ID dané budovy a proměnnou nová výška budovy. Výsledný slovník bude obsahovat ID všech budov a jejich nové výšky. Je tomu tak z důvodu, že výsledek optimalizační úlohy uvádí výšky všech výsledných agregátů (i kdyby agregátem měla být jedna budova a výška těla budovy by se nezměnila). A v omezujících podmínkách optimalizační úlohy je definováno, že každá budova musí náležet pouze jednomu centru (i za podmínek že by měla náležet sama sobě).

`create_dict_new_roof_height (center_matrix, height_roof_center)`

Funguje obdobně jako předchozí funkce s tím rozdílem, že vstupním parametrem je výška střechy výsledného agregátu. Skript parsuje hodnotu klíče slovníku center výšek střechy agregátů tak, aby bylo odděleno ID budovy. Další rozdíl oproti výškám těl agregátů je, že budovy, jenž mají rovný typ střešní plochy mají definovanou výšku střechy 0. Výsledkem této funkce je slovník, jehož klíčem je ID dané budovy a proměnnou nově definovaná výška střechy. Výstupní slovník neobsahuje hodnoty budov, které mají plochou střechu.

`calculate_new_z_cords_body (new_body_heights, z_footprints)`

Ze slovníku nových výšek agregovaných těl budov vytváří nové Z souřadnice maximální výšky těla budovy. Navíc mění formát ID budov, které jsou výstupem optimalizační úlohy, na původní formát. Konkrétně nahrazuje podtržítka za pomlčky, více o změně formátu vstupních dat v optimalizační úloze viz kapitola *5.3*. Původně měla být *Z* souřadnice výšky budovy vypočítána tak, že by byl proveden součet Z souřadnice půdorysu budovy a nové výšky. Avšak vzhledem k tomu, že byly všechny modely pro tuto práci vytvářeny uměle a výška *Z* souřadnice je s minimálními odchylkami 0, byla výška půdorysu všech budov definována jako 0. Výstupem je slovník, jehož klíčem je ID budovy a proměnnou nová *Z* souřadnice výšky těla budovy. Počet položek ve slovníku odpovídá počtu budov v datovém modelu.

`calculate_new_z_cords_roof (new_roof_heights, z_max_body, z_max_roof)`

Ze slovníku agregovaných výšek střešní části budovy vytváří nové *Z* souřadnice. Ty jsou vytvářeny tak, že je proveden součet nově vzniklé výšky těla budovy z předchozího skriptu a výšky střešní části. Pokud je v původním slovníku hodnota proměnné 0, je danému ID definována proměnná 0 a položka je zapsána do výstupního slovníku. Je změněn formát ID budov, které jsou výstupem optimalizační úlohy, na původní formát. Výstupem je slovník, jehož klíčem je ID budov a proměnnou nová *Z* souřadnice maximální výšky střechy. Počet položek ve slovníku odpovídá počtu budov v datovém modelu.

Druhou částí skriptu pro vizualizaci výsledků optimalizační úlohy je úprava *OBJ* formátu. Ta je provedena tak, že jsou původní *Z* souřadnice dané budovy nahrazeny novými *Z* souřadnicemi dané budovy.

Jsou známy: původní *Z* souřadnice výšky těla budovy, původní *Z* souřadnice výšky střešní části budovy, původní *Z* souřadnice půdorysu budovy. Dále jsou známy nové *Z* souřadnice výšky těla a střešní části budovy. Všechna data jsou uložena ve formě slovníků. Klíčem je ID dané budovy a proměnnou daná *Z* souřadnice.

Vstupní *OBJ* formáty datových modelů vytvořených pro tuto práci mají jednoduchou strukturu. Obsahují pouze tři druhy prvků:

- *Vertex* – [v] na tomto řádku jsou uvedeny *X, Y, Z* souřadnice daného vertexu, přičemž závisí na pořadí řádku v OBJ formátu

- *Faces* – [f] – na řádku jsou uvedeny čísla vertexů, které tvoří plochu daného objektu. Co řádek to jedna plocha
- *Object name* – [o] – řádek uvádí název daného objektu, který je tvořen několika plochami

Obrázek č. 12 znázorňuje strukturu zápisu jedné budovy ve formátu OBJ.

```
v 0.0 0.0 15.3499999843
v 7.25999999259 0.0 15.3499999843
v 7.25999999259 9.81999998998 15.3499999843
v 0.0 9.81999998998 15.3499999843
v 0.0 0.0 0.0
v 7.25999999259 0.0 0.0
v 7.25999999259 9.81999998998 0.0
v 0.0 9.81999998998 0.0

o ['556bba9a-2cdb-450b-ae7a-98cf05a84b6c']
f 1 2 3 4
f 5 6 2 1
f 6 7 3 2
f 7 8 4 3
f 8 5 1 4
f 5 8 7 6
```

Obrázek č. 12: Struktura OBJ formátu jedna budova

Zdroj: vlastní tvorba, konverze formátu z CityGML na OBJ (Biljecki a kol. 2015a)

`load_obj(filename)`

Načte datový soubor formátu OBJ a vytvoří samostatný list s hlavičkou OBJ formátu. Následně list vertexů a slovník, jehož klíčem jsou ID objektu a proměnnými hodnoty *faces*.

`change_height (obj, vertices, old, new)`

U vertexu konkrétní budovy změní starou výšku za novou. Je součástí funkce `modify_objects`.

modify_objects (objects, vertices, fdict, tdict)

Mění starou výšku za novou jen pokud je stará i nová výška nenulovou hodnotou a pokud najde shodu s ID v datech z OBJ souboru.

change_height_footprints (obj, vertices, old):

Změní všechny *Z* souřadnice půdorysu na 0. Je součástí funkce *modify_objects_footprints*.

modify_objects_footprints (objects, vertices, fdict)

Mění *Z* souřadnici půdorysu na 0. Jen pokud je stará výška nenulovou hodnotou a pokud najde shodu ID v datech z OBJ souboru.

save_obj (header, vertices, objects, filename, centers)

Uloží změny do OBJ souboru s názvem: *„out.obj“*

Výsledkem vizualizačního skriptu je soubor formátu OBJ, který obsahuje model budov se změněnými výškami těla budovy a střešní části. Také dochází k „zarovnání“ půdorysu, kdy jsou všechny *Z* souřadnice půdorysu nastaveny na 0 a od této hodnoty se následně počítá výška těla a střešní části budovy.

Takto změněný OBJ soubor neobsahuje informace o tom, které budovy patří do jednoho agregátu. Pouze je budovám změněna výška. Z tohoto důvodu byla do skriptu přidána funkce, která rozčleňuje budovy do jednotlivých agregačních skupin.

create_center_groups(center_matrix)

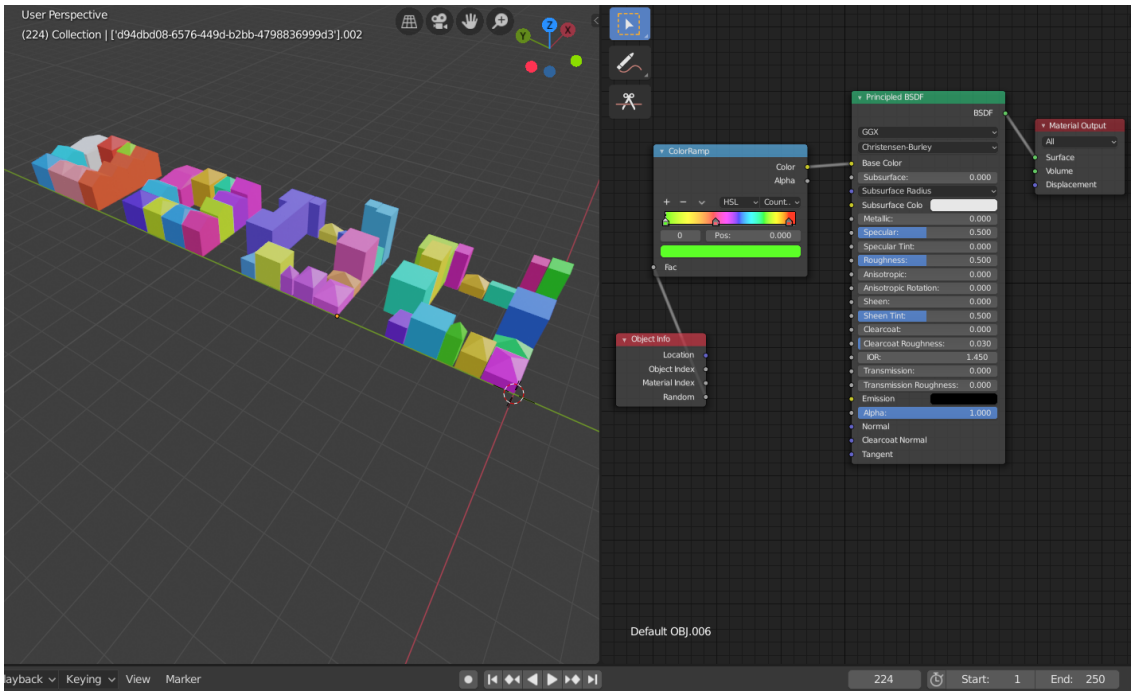
Ze slovníku `center_matrix` vybere pouze položky, které obsahují nenulové hodnoty. Jedna položka ve slovníku je pak následujícího tvaru:

```
"center_matrix_('03f93f78_06c3_4ade_8e1b_88048bd44518',_'03f93f78_06c3_4ade_8e1b_88048bd44518')": 1.0
```

První hodnota v závorce udává ID centra agregátu a druhá hodnota je ID budovy, která má být agregována. Položka je parsována tak, aby vznikl nový slovník, jehož klíčem bude ID budovy, která má být agregována a hodnotou ID centra agregátu. Následně je funkce *save_obj* upravena tak, aby byl za řádkem udávajícím název objektu řádek, který charakterizuje příslušnost budovy k danému agregátu. Řádek jo značen písmenem *g (group)* – což v *OBJ* specifikaci znamená název skupiny.

Následně jsou takto vzniklé *OBJ* soubory importovány do programu *Blender*, pomocí importního nástroje. Importní nástroj mimo jiné umožňuje definovat souřadnice v souřadném systému, což umožňuje správné natočení modelu bloku. Dále je možné určit, zda se mají objekty importovat jako jednotlivé budovy či jako skupiny agregovaných budov. Pro výsledné modely budov je zvolena varianta importu skupiny agregátů.

Pomocí *Texture node editoru* v programu *Blender* byla vytvořena funkce, která automaticky obarví jednotlivé budovy – u vstupních dat a jednotlivé agregáty – u výstupních dat, viz obrázek č. 13.

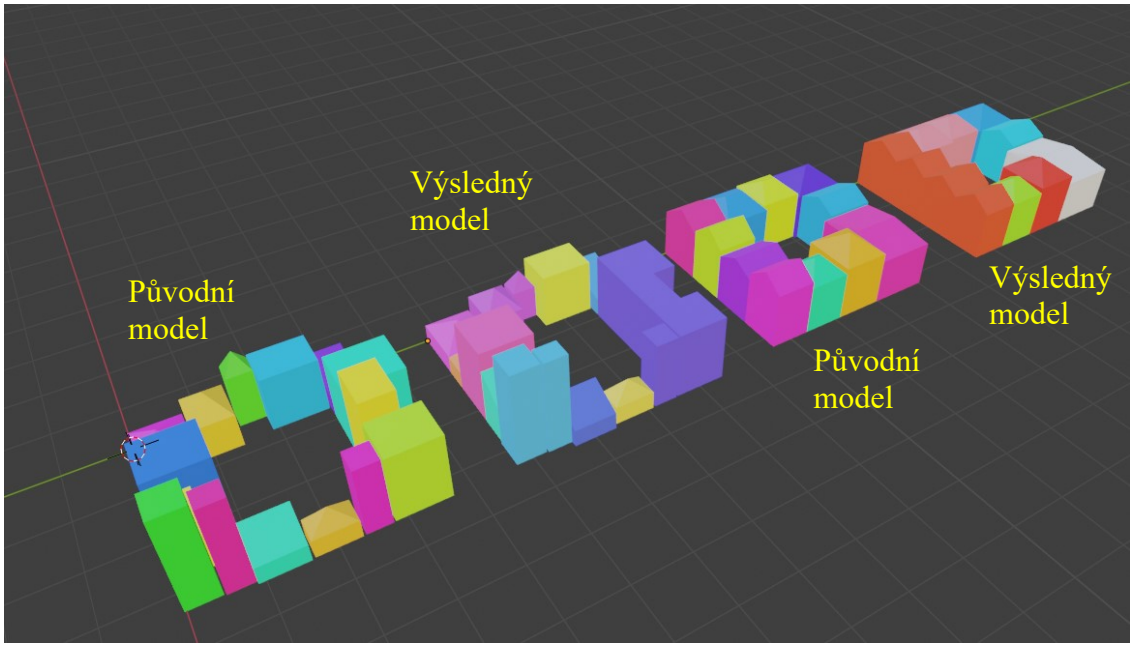


Obrázek č. 13: Automatizace barevného rozlišení budov nebo agregačních bloků v programu Blender

Zdroj: vlastní tvorba

Vlastností vizualizace v programu *Blender* je to, že některé budovy, které jsou bezprostředně blízko sebe, vypadají již jako spojené agregáty. Pokud mezi agregáty existovala mezera, budova se vizuálně nespojí. Budova však může patřit k jednomu agregátu, proto je klíčové obarvení budov. Sousední budovy, které mají stejnou barvu patří do jednoho agregátu. Podobnost barev nesousedních budov je čistě náhodná a souvisí s automatickým obarvením jednotlivých budov či agregátů.

Výstupem celé vizualizační kapitoly jsou barevně odlišené skupiny budov, které mají tvořit agregační celky. Budovy mají v daném celku stejnou výšku střešní plochy a zároveň i stejnou výšku těla budovy, viz obrázek č. 14.



Obrázek č. 14: Výstup vizualizace, barevně rozlišené agregáty ve výsledných modelech, stejné výšky střešní části i těl budovy

Zdroj: vlastní tvorba

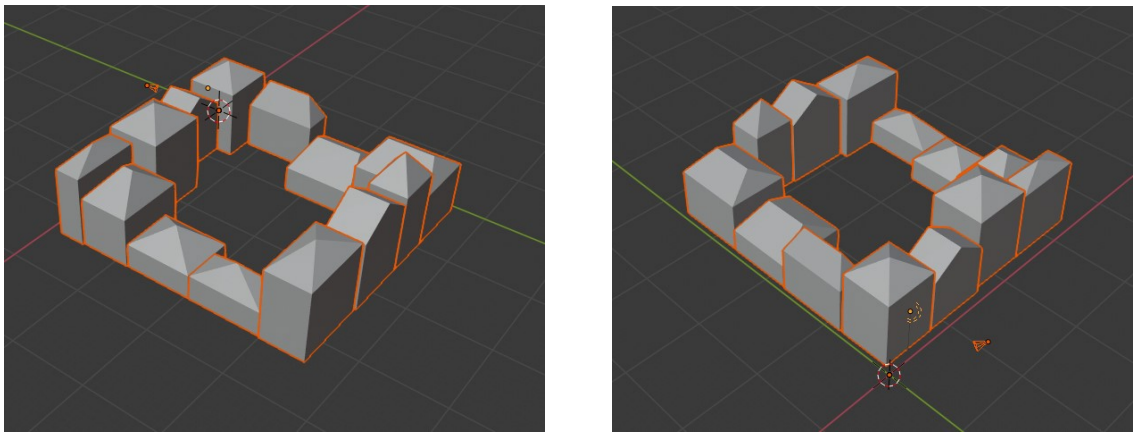
7. Experimentální výsledky

Výsledky lze rozdělit na tři části. První částí je příprava geometrických dat, tedy jednotlivých budov. Příprava probíhá pomocí procedurálního modelování. Následuje úprava budov, aby vznikly souvislé bloky. Dále je prováděn výpočet a odečtení základních parametrů budovy.

Druhou částí je výstup optimalizační úlohy v podobě hodnot původně neznámých proměnných. Ty udávají změnu výšky či objemu daného tělesa a jeho příslušnost k danému agregátu.

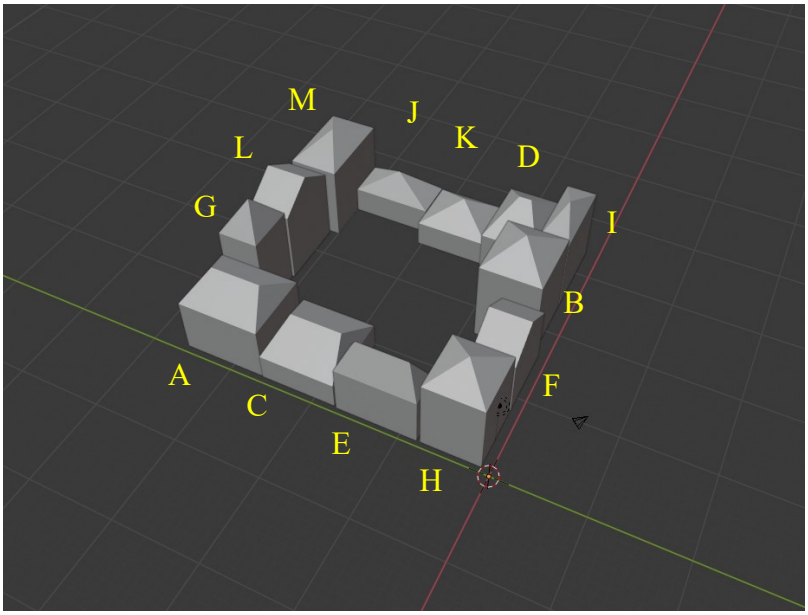
Třetí částí je pak vizualizace výsledku v podobě barevně rozlišených agregovaných celků s upravenou výškou těla agregátů a střešní části tak, aby byly výšky agregátů jednotné.

Níže bude popsáno a vizualizováno několik bloků budov se změněnými parametry optimalizační úlohy. Výsledky optimalizační úlohy obsahují hodnoty všech původně neznámých proměnných. Některé z nich však nemají pro další práci význam, ale jsou důležité pro správný chod optimalizace. Takovými proměnnými je suma pozitivního toku sousedství *fa* a ukazatel pozitivního toku na hraně *Fa*. Z důvodu větší přehlednosti výsledků nejsou tyto hodnoty v následujících přehledech výsledků uvedeny. Důležitou proměnou jsou hodnoty centrální matice a také výšky výsledných agregátů – objektů. Dále také objemové změny v absolutní hodnotě. Na obrázku č. 15. je zobrazen model prvního testovacího bloku, skládající se z třinácti budov. Model je zobrazen ze dvou úhlů pohledu.



Obrázek č. 15: Model bloku budov před vstupem do optimalizační úlohy
Zdroj: vlastní tvorba

Z důvodu větší přehlednosti, byly ID budov převedeny na písmena abecedy A – M. Na obrázku č. 16 je zobrazen testovaný model budov, s názvy ID. Tabulka č. 4 zobrazuje hodnoty volitelných parametrů. Příloha č. 1 pak výsledky optimalizační úlohy v podobě matice sousednosti. Pokud je v tabulce uvedena hodnota 1, budova patří do daného agregátu. Dále příloha č. 1 obsahuje výšky těla a střešní části centra agregátů. Kde je u výšek uvedena hodnota 0, výška budovy je shodná s výškou centra agregátu, do kterého budova spadá. Výjimkou jsou ploché střechy. U plochých střech je výška střechy vždy definována jako 0. V příloze č. 2 jsou maticově uvedeny hodnoty objemových změn těla a střešní části jednotlivých budov.



Obrázek č. 16: Tetovaný model bloku budov s názvy ID budov

Zdroj: vlastní tvorba

Proměnná	Hodnota
epsilon_roof_type	0
epsilon_roof_orientation	1
epsilon_roof_height	5
epsilon_height	10
body_volume_change_weight	0,001
roof_volume_change_weight	0,001

Tabulka č. 4: Hodnoty volitelných proměnných

Zdroj: vlastní tvorba

Hodnota objektivní funkce této optimalizační úlohy je 8,698. Tato hodnota byla získána výpočtem skriptu. Jde o součet všech objektů v agregovaném modelu budov a všech objemových změn. Objektem je myšlen nově vzniklý agregát budov, nebo může být za objekt považována samostatná budova, která se nespojila s žádnou jinou budovou. Z hodnot tabulky centrální matice uvedené v příloze č. 1 lze zjistit, že suma agregátů je 8. Z toho vyplývá, že hodnota 0,698 udává objemové změny v absolutní hodnotě. To lze ověřit součtem všech objemových změn uvedených v příloze č. 2. Součet objemových změn je nutné vydělit váhou objemové změny. V tomto případě hodnotou 0,001. Součet objemových změn uvedených v příloze č. 2 je shodný s hodnotou objemové změny, získané z výsledku objektivní funkce a jeho výsledek je 697,73.

Součet objemových změn v absolutní hodnotě lze rozdělit na objemové změny těla budovy, ty činí 661,41 a objemové změny střešní části, ty činí 36,32. Hodnotu součtu objemové změny lze porovnat se součtem všech objemů v původním modelu, ten je 5182,69. Součet objemů všech celků v nově vzniklém modelu je 5063,79. Objemové změny i sumy původního objemu jsou uváděny v bezrozměrných jednotkách. Je tomu tak z důvodu, že modely byly vytvářeny uměle pomocí procedurálního modelování a jednotky neodpovídají délkovým či objemovým hodnotám v reálném světě. Tyto výsledky lze shrnout do přehledné tabulky, viz tabulka č. 5.

Výsledky optimalizace – testovací model 1	
WB= 0,001 WR=0,001	
hodnota objektivní funkce	8,70
počet objektů	8
suma objemových změn těla budovy (abs)	661,41
suma objemových změn střešní části (abs)	36,32
suma objemových změn celkem (abs)	697,73
součet objemu celého modelu	5063,79

Tabulka č. 5: Přehled výsledků optimalizační úlohy

Zdroj: vlastní tvorba

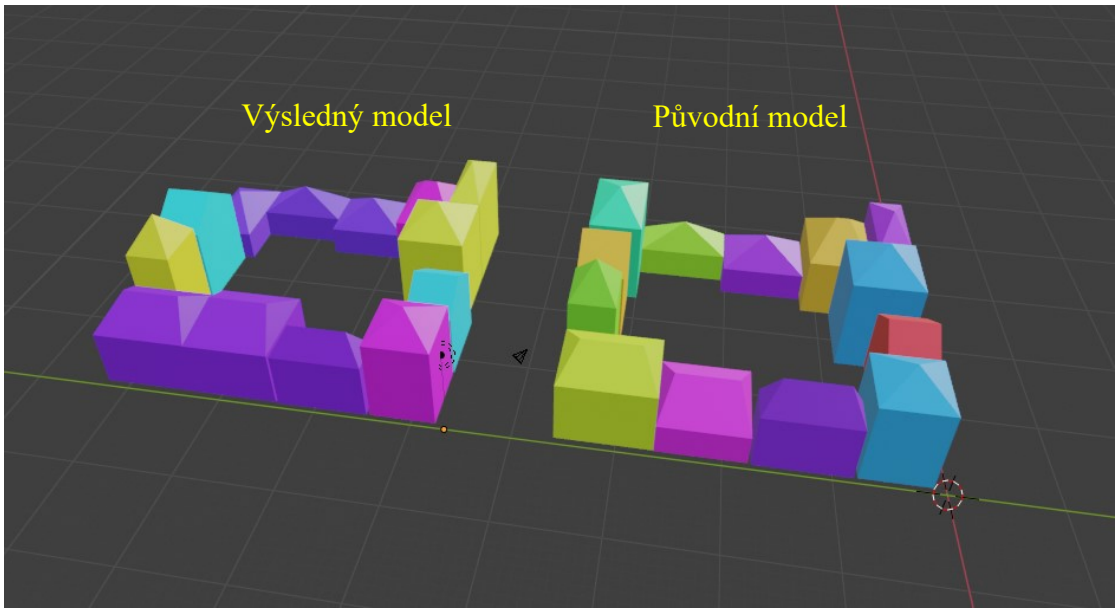
U hodnot objemových zněn uvedených v příloze č. 2 je nutné připomenout, že jde o absolutní hodnotu objemových změn. Výsledek optimalizační úlohy tedy neobsahuje přímou informaci o tom, zda jde o objemovou změnu zvětšení či zmenšení budovy. Tuto informaci lze získat porovnáním výšky budovy před vstupem a po vstupu do

optimalizační úlohy. A to z toho důvodu, že optimalizační úloha je nastavena tak, že jsou měněny pouze výšky budov. Informace o změnách výšek jsou uvedeny v tabulce č. 6.

Změna výšek jednotlivých budov - tělo budovy													
ID budov	A	B	C	D	E	F	G	H	I	J	K	L	M
Původní výška	6,82	10,29	3,26	6,58	6,10	6,98	9,99	8,99	9,29	3,33	3,24	9,09	9,41
Nová výška	6,10	10,29	6,10	6,58	6,10	6,98	9,99	9,00	10,29	3,33	3,33	9,09	3,33
Změna výšek jednotlivých budov - střešní část budovy													
ID budov	A	B	C	D	E	F	G	H	I	J	K	L	M
Původní výška	2,52	2,66	2,79	3,25	3,40	2,65	3,55	2,31	1,99	2,80	2,80	2,85	2,22
Nová výška	2,80	2,66	2,80	3,25	2,80	2,65	3,55	2,31	2,66	2,80	2,80	2,85	2,80

Tabulka č. 6: Výšky jednotlivých budov před a po výsledku optimalizační úlohy
Zdroj: vlastní tvorba

Hodnoty nových výšek byly získány modifikací tabulek výšek center agregátů uvedených v příloze č.1. Nulové hodnoty byly nahrazeny výškou agregátu, do kterého daná budova spadá. Z tabulky č. 6 je patrné, že některé budovy svoji výšku zmenšily, např. budova M a jiné svou výšku zvětšily, např. budova C. U některých budov zase zůstala výška konstantní. Obrázek č. 17 graficky znázorňuje výsledek výše popsané optimalizační úlohy. Jednotlivé agregáty jsou barevně rozlišeny a mají konstantní výšku těla budovy i střešní části.



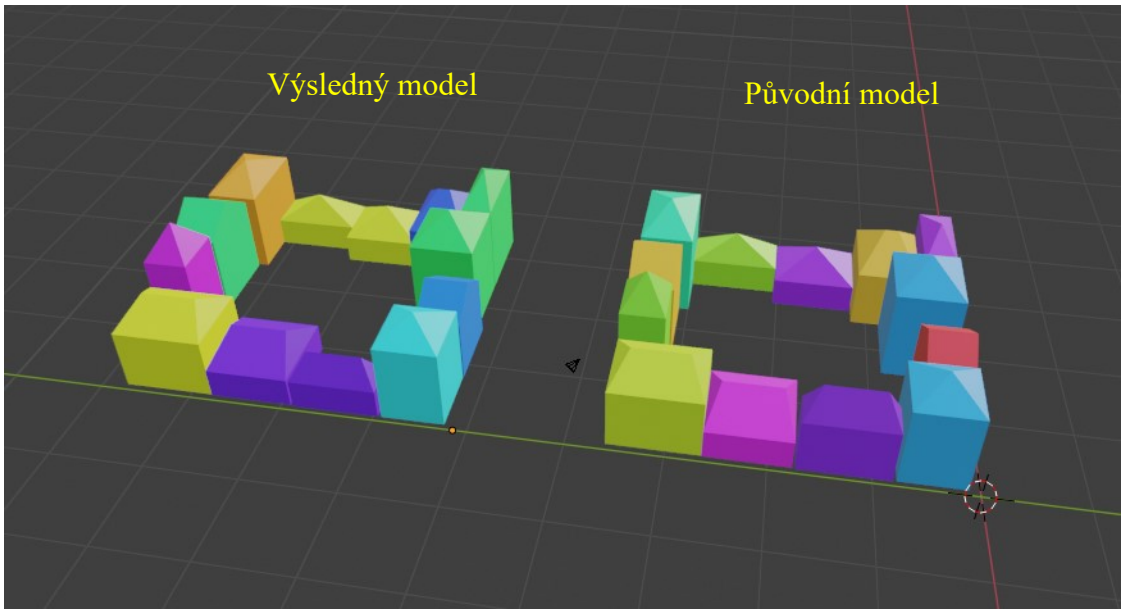
Obrázek č. 17: Grafický výsledek optimalizační úlohy vlevo vs původní model vpravo
Zdroj: vlastní tvorba

Z obrázku č. 17 je patrné, že proměnné *WB* a *WR* byly nastaveny tak, že byl minimalizován výsledný počet agregátů. Další agregování budov limituje podmínka *epsilon_roof_type*, která je nastavená tak, že jsou agregovány pouze budovy se stejným typem střechy. V tomto případě nedojde při dalším snížení parametrů *WB* a *WR* k žádným velkým změnám. I z pohledu změny výšek jednotlivých agregátů. K agregaci dalších budov by došlo v případě změny nastavení hodnoty *epsilon_roof_type* tak, aby bylo povoleno agregovat budovy, které mají rozdílný typ střechy. Jak již bylo zmíněno v předchozích kapitolách, omezující podmínky, které obsahují parametry epsilon, určují, které budovy je možné agregovat a které ne. Pokud by uživatel chtěl docílit agregace všech budov v jeden celek, bylo by nutné uvolnit podmínky tak, aby bylo možné agregovat budovy s různým typem střešní plochy, budovy s libovolným rozdílem výšek a libovolnou rotací vrcholové střešní linie. V praxi to znamená zvětšit interval epsilon, viz kapitola č. 5.4.

Takto podrobná data v podobě tabulek uvedených v příloze 1 a 2 nelze z důvodu velkého počtu možných výsledků uvádět pro každý model budov. Hodnoty výsledků se mění při změně hodnot volitelných proměnných. V následujících modelech bude proto uvedena pouze hodnota objektivní funkce, počet výsledných celků v agregovaném modelu budov, suma absolutní hodnoty objemových změn těla a střešní části budovy a jejich součet. Výše zmíněný model třinácti budov byl znovu testován při změněných parametrech *WB a WR* viz obrázek č. 18 a tabulka č. 7.

Výsledky optimalizace – testovací model 1	
WB= 0,005 WR=0,01	
hodnota objektivní funkce	10,83
počet objektů	10
suma objemových změn těla budovy (abs)	133,95
suma objemových změn střešní části (abs)	16,25
suma objemových změn celkem (abs)	150,20
součet objemu celého modelu	5110,72

Tabulka č. 7: nastavení parametrů objemových změn a výsledky optimalizace
Zdroj: vlastní tvorba



Obrázek č. 18: Grafický výsledek optimalizační úlohy vlevo vs původní model vpravo, hodnoty parametrů WB= 0,005 WR=0,01

Zdroj: vlastní tvorba

Dále byly testovány i jiné modely bloků budov. Výsledky optimalizační úlohy nad jednotlivými modely jsou zobrazeny v přílohách č. 3 až 10. V každé z příloh je graficky znázorněn původní model bloku. Je uveden počet budov a součet objemů všech budov v původním modelu. Dále také parametry epsilon, které jsou pro optimalizace modelů konstantní. Volitelné proměnné *WB a WR* jsou v optimalizačních úlohách měněny, a proto je u každého výsledku uvedena i hodnota této proměnné.

Výsledky optimalizace uvádí: výsledek objektivní funkce, počet objektů v modelu, sumu objemových změn těla a střešní části budovy v absolutní hodnotě a jejich součet. Jako poslední je v tabulce uveden součet objemů všech celků modelu budov po optimalizaci. Následně je graficky znázorněn výsledek optimalizace.

V příloze č. 3 je zobrazen výsledek optimalizační úlohy, kdy parametry epsilon působí jako omezení agregace některých budov do bloků. Příklad také ukazuje na možnost nastavení rozdílných hodnot parametrů *WB* a *WR* a jaké má nastavení vliv na výsledek optimalizace. V příloze č. 4 jsou pak zobrazeny dva extrémní případy. Hodnoty parametrů epsilon jsou nastaveny tak, aby neomezovali agregaci budov do bloků. Záleží tedy pouze na nastavení parametrů *WB* a *WR*. Pokud jsou parametry objemových změn nastaveny na velmi nízkou hodnotu, je provedena agregace všech budov do jednoho celku. V opačném případě, pokud se parametry objemových změn blíží jedné, není

agregována ani jedna budova. Příloha č. 5 zobrazuje možnost agregace budov do bloků v případě, že je povoleno agregovat některé typy střech dohromady. Parametry typů střech budov jsou normalizovány na hodnoty celých čísel v rozmezí 1 až 5, viz kapitola 5.2. Parametr epsilon je nastaven na hodnotu 2, to znamená, že mohou být agregovány budovy, pouze pokud je rozdíl normalizovaných parametrů střech menší než dva, viz kapitola 5.2 a 5.4. Z výsledků optimalizace tohoto modelu je patrné, že minimalizace objemové změny v absolutní hodnotě, ještě neznamená minimalizaci objemové změny nově vzniklého součtu objemů budov vůči původnímu součtu objemů budov. Tato situace není častá, ale je potřeba počítat s tím, že může nastat.

Optimalizační úloha může být aplikována, na různé druhy bloků, nemusí jít vždy o uzavřené celky, viz příloha č. 6. V příloze č. 7 je zobrazeno „tvrdé“ omezení výšky *epsilon_height*, kdy je zakázáno agregovat budovy, jejichž rozdíl výšek je větší než tři. Další možnosti nastavení parametrů optimalizace jsou uvedeny v příloze č. 8. Příloha č. 9 a 10 zobrazuje testování modelu, který na vstupu obsahuje více bloků.

8. Diskuze

Z výsledků práce je patrné, že metoda matematické optimalizace může sloužit jako nástroj ke spojení jednotlivých budov LOD2 v bloku do agregačních celků. Díky množství nastavitelných parametrů, může vyhovět potřebám široké škále uživatelů.

Při hledání vhodných vstupních dat bylo z důvodů velkého množství topologických a geometrických chyb v reálných modelech přistoupeno k vytvoření vlastních dat pomocí procedurálního modelování. Výhodou procedurálního modelování je, že je model získán rovnou v šestnácti úrovních detailu. Neobsahuje topologické a geometrické chyby. Uživatel si může navíc tvorbu jednotlivých modelů přizpůsobit vlastním potřebám. To je jeden z důvodů, proč mohou modely sloužit i k mnoha dalším výzkumům.

Byly vybrány dva datové formáty vhodné pro uložení 3D modelů bloků budov. *CityGML* je vhodný z důvodu uložení dodatečných informací o jednotlivých budovách. *Wavefront* je vhodný k vizualizaci modelů. Bylo zjištěno, že formát *CityGML*, obecně uvedený jako standard pro ukládání 3D modelů měst, není podporován řadou volně dostupných vizualizačních softwarů. Pro vizualizaci modelů je výhodnější převést model do formátu *Wavefront*.

Optimalizační úloha byla nastavena tak, aby byl minimalizován počet nově vzniklých objektů. Zároveň je možné nastavit pomocí volitelných parametrů, jak velký vliv bude mít minimalizace objemových změn na výsledek. Dále je možné nastavit „tvrdá“ omezení, které budovy již neagregovat. Výsledek optimalizační úlohy uvádí, jaké budovy k sobě agregovat a jak se po agregaci změní výška a objem v absolutní hodnotě jednotlivých budov.

Úloha optimalizační funkce pokrývá základní parametry budovy LOD2. Je však možné přidat řadu dalších rozšíření v podobě omezujících podmínek. Nebo rozšířit stávající omezující podmínky, těmi mohou být: agregovat budovy s podobným sklonem střešní plochy, či agregovat budovy s podobným obsahem střešní plochy (tyto informace jsou důležité pro instalaci solárních panelů). Dále může být optimalizační úloha rozšířena o možnost agregace stejných typů střech, které mají vůči sobě specifickou polohu natočení střešní plochy. To by bylo vhodné u šikmých typů střech, kdy existuje 16 možností vzájemného natočení dvou budov.

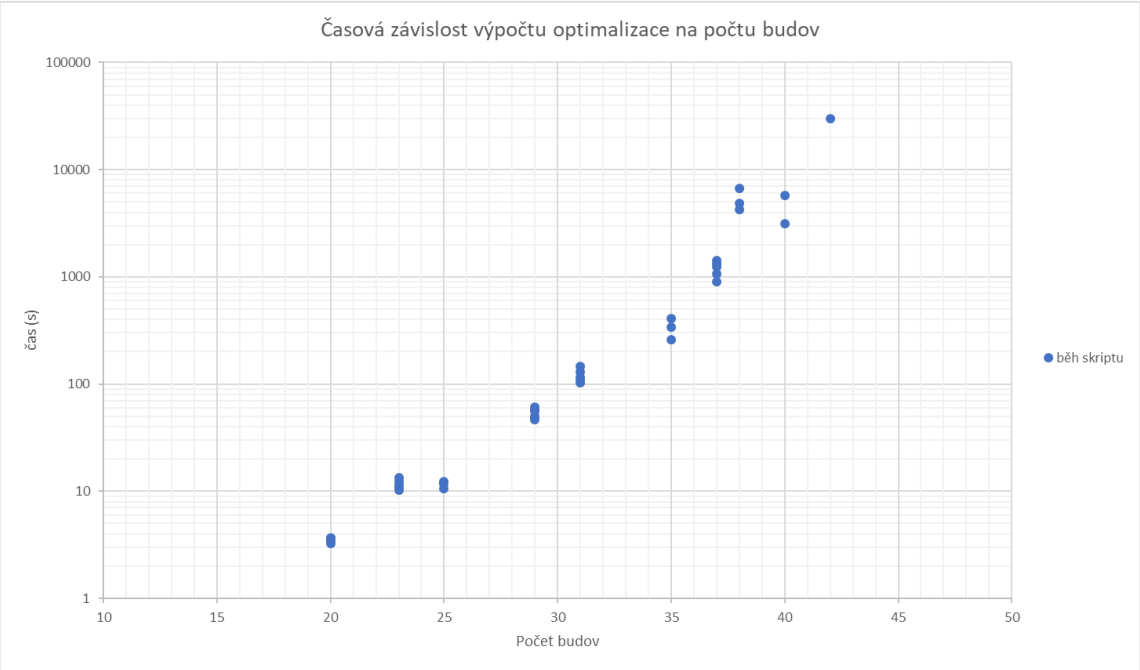
Všechny skripty vytvořené v této práci byly napsány v jazyce *Python*. Pro každý skript byly implementovány vhodné knihovny. Při tvoření optimalizační úlohy bylo

zjištěno omezení, kdy knihovna *Pulp* využívaná pro účel lineárního programování nepodporuje matematický operátor násobení mezi dvěma neznámými proměnnými.

Následně byl vytvořen skript vizualizace výsledků optimalizace. Ta je provedena změnou výšek jednotlivých budov a jejich střešních částí tak, aby byly výšky v jednom bloku konstantní. Dále jsou od sebe barevně odděleny jednotlivé objekty. Práce již neřeší úpravu, agregaci či eliminaci jednotlivých ploch, které formulovaly původní budovy.

Časová náročnost výpočtu optimalizační úlohy se liší v závislosti na: výpočetní technice, nastavení parametrů a vstupních datech. U vstupních dat velmi záleží na počtu budov a také na diferenci jednotlivých parametrů budovy, které vstupují do optimalizační úlohy. Těmi jsou: typ střešní plochy, natočení budovy a rozdíl výšek agregovaných budov. Obecně platí, že čím „striktnější“ je parametr epsilon (např. slučování střech pouze se stejným typem), tím méně možností na agregaci vzniká a tím je program rychlejší. V tomto případě ale velmi záleží na vstupních datech, pokud je vstupním blokem skupina budov se stejným typem střechy, omezení *epsilon_roof_type* výpočetní náročnost neovlivní. Váha objemových změn také ovlivňuje rychlost výpočtu optimalizace, čím je váha nižší, tím je výpočetní náročnost nižší a časová náročnost výpočtu modelu se zmenšuje.

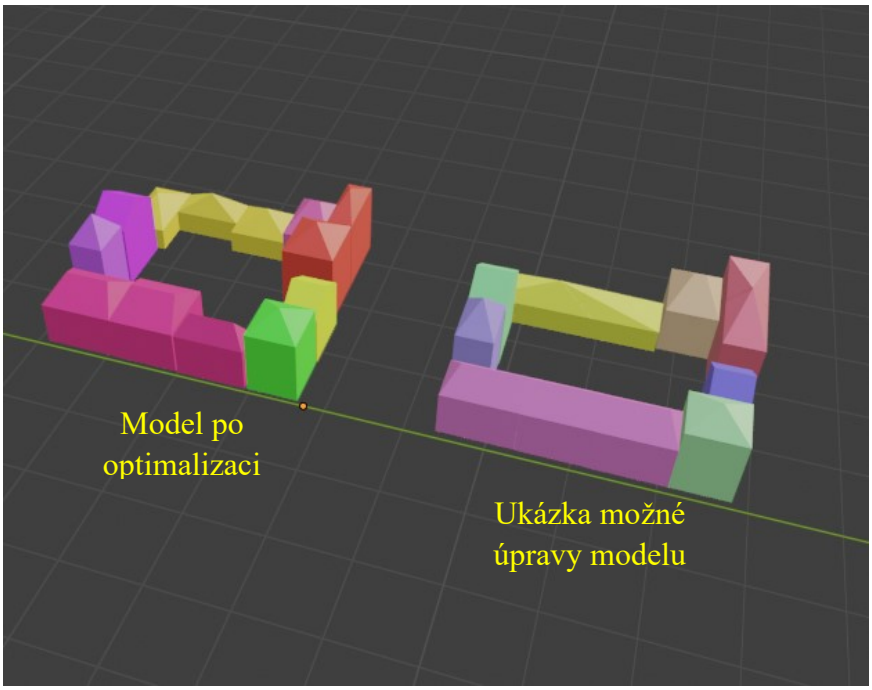
Dalším faktorem ovlivňujícím náročnost je počet budov ve vstupním modelu. Pro přehled časové náročnosti optimalizační úlohy byl vytvořen graf č.1. Ten značí závislost počtu budov na čase výpočtu optimalizace. Testován byl jeden model složený z více bloků budov, ze kterého byly postupně odstraňovány jednotlivé budovy. Nastavení parametrů bylo po celou dobu testování konstantní. Testování probíhalo na jednom počítači.



Graf č. 1: časová náročnost běhu optimalizačního skriptu
(zdroj: vlastní tvorba)

Z grafu č. 1 vyplývá, že pro počet budov 38 a více se čas výpočtu optimalizace razantně zvyšuje. Pro 42 budov trvá výpočet více než 8 hodin. K dosažení rychlých výsledků je lepší optimalizovat každý blok budov zvlášť, nebo rozdělit vstupní data na několik menších modelů.

Další vědecká činnost by se mohla zabývat metodami použití výsledků optimalizační úlohy pro úpravu geometrických dat vstupních modelů. Jako je zarovnání půdorysu, odstranění přebytečných ploch, či možnostmi vizuální agregace střech. V případě agregace více střešních typů, pak metodami sjednocení střech. Pro další vědeckou činnost je výhodné, že pomocí výsledků optimalizační úlohy lze dopočítat objemy jednotlivých nově vzniklých agregátů. Bude tedy možné upravit půdorys budov tak, aby se minimalizovala změna objemu výsledného agregátu. Na obrázku č. 19 je zobrazen návrh, jak by taková změna mohla vypadat. Vstupem je testovací model 1.



Obrázek č. 19: Návrh úpravy půdorysu agregátů po optimalizaci
Zdroj: vlastní tvorba

V dostupných zdrojích prozatím nebyla nalezena publikace, která by řešila agregaci budov LOD2 do bloků s důrazem na zachování určitých charakteristik budov. Zároveň práce dokázala, že je metoda matematické optimalizace schopna řešit problém agregace budov LOD2. Další výhodou je, že navrženou optimalizační úlohu je možné rozšiřovat dle potřeb výzkumu.

9. Závěr

V této práci byla navržena metoda agregace jednotlivých budov v bloku do spojitých celků ve 3D prostoru. Výchozím modelem byly budovy LOD2. Pomocí metody matematické optimalizace byly budovy v jednotlivých blocích agregovány na základě podobnostních charakteristik. Budovy byly agregovány tak, aby byl minimalizován počet nově vzniklých agregátů a zároveň, aby byla minimalizována objemová změna budovy. Z dostupných publikací není prozatím znám žádný výzkum, který by řešil agregaci budov v blocích LOD2 s důrazem na zachování určitých charakteristik budovy.

Z rešerše této práce bylo zjištěno, že na problém agregace budov bylo prozatím pohlíženo pouze z pohledu geometrie a vizuálního vjemu výsledného agregátu. V mnoha případech byly agregací zanedbávány základní charakteristiky budovy. Těmi jsou například výška a objem budov, nebo typ střešní plochy. Dále bylo zjištěno, že problémy agregace bloku budov do souvislých celků a problémy generalizace střešních ploch byly řešeny separátně.

Při výběru dat vhodných pro tuto práci, bylo zjištěno, že volně dostupná data není možné použít. Současné modely reálných měst obsahují řadu topologických a geometrických chyb, které nejsou vhodné pro modelování objemového tělesa. Z tohoto důvodu byly generovány vlastní budovy LOD2 pomocí procedurálního modelování. Budovy byly upraveny do souvislých bloků. Následně byly řešeny datové formáty vhodné pro uložení 3D modelů. Byly vybrány dva datové formáty *CityGML (.gml/.xml)* a *Wavefront (.obj)*. Formát *CityGML* je vhodný pro ukládání podrobných informací o jednotlivých budovách. *Wavefront* je zase vhodný pro vizualizaci, protože je jednodušší na čtení a podporuje ho řada vizualizačních softwarů. Byl vytvořen skript, který počítá základní parametry budovy. Ty jsou potřeba jako vstup do optimalizační úlohy a následně pro vizualizaci výsledku.

Při hledání vhodné agregační metody byl kladen důraz na robustní řešení problému. Takovým řešením je metoda matematické optimalizace, přesněji lineárního programování. Práce vycházela z článku Guercke a kol. (2011), který řeší agregaci bloků budov LOD1 pomocí metody lineárního programování. Byla zreprodukována část optimalizační úlohy. Následně byla navržena vlastní metoda rozšíření optimalizační úlohy o parametry budovy LOD2. Tím je zejména typ střešní plochy a její natočení. Dále byla přidána objemová změna střešní části budovy a parametry omezující agregaci budov

s velkým rozdílem výšek. Optimalizační úloha byla nastavena tak, aby byl minimalizován počet center agregátů, přičemž každá budova může být centrem. Zároveň optimalizační úloha řeší minimalizaci objemové změny budovy po agregaci. Byly zavedeny parametry *WB* a *WR*, které určují váhu objemové změny vůči minimalizaci center agregátu. Je na potřebě uživatele, zda upřednostní minimalizaci center agregátu či objemové změny. Objem budovy je měněn pouze změnou výšky dané budovy. Dále byly zavedeny omezující podmínky, které v závislosti na nastavení parametrů vylučují agregaci některých budov. Agregace jednotlivých budov může být vyloučena na základě velkého rozdílu výšky sousedních budov, rozdílné orientace střešní linie budov, či na základě rozdílného typu střech.

Byla provedena vizualizace optimalizační úlohy. Byl vytvořen skript, pomocí kterého byla změněna výška původním blokům dle výsledku optimalizační úlohy a jednotlivé budovy v bloku byly barevně rozčleněny do agregačních celků, rovněž dle výsledku optimalizace.

Z experimentálních výsledků je patrné, že úloha matematické optimalizace, může sloužit jako nástroj agregace budov. Za správně definovaných podmínek a vhodně nastavených parametrů může být nalezeno nejvhodnější řešení. Z jedné strany z pohledu uživatele a jeho potřeby použití zjednodušeného modelu pro další analýzu. A z druhé strany optimalizační úloha hledá minimum dané funkce, tím pádem najde to nejlepší řešení ze všech možností. V tom je výhoda oproti heuristickému řešení problému, kde může být optimální řešení nechtěně vyloučeno.

V neposlední řadě práce poukazuje na možnosti rozšíření optimalizační úlohy a následné vizualizace budov. Optimalizační úlohu lze rozšířit o další omezující podmínky. Například agregovat pouze budovy s podobným sklonem střešní plochy atd. Z pohledu vizualizace je možné z výsledku změny výšek určit objem celého agregátu a následně změnit půdorys budovy tak, aby tvořil jednoduchý tvar, např. obdélník a zároveň aby, byla zachována objemová změna agregátu.

Seznam použité literatury

BARTSCH, H. J. (1983). Matematické vzorce. Praha: SNTL – Nakladatelství technické literatury.

BILJECKI, F. a kol. (2014a): Formalisation of the level of detail in 3D city modeling. Computers, Environmentand Urban Systems, 48 (16), 1–15.

BILJECKI, F., LEDOUX, H., STOTER, J. (2014b): Height references of CityGML LOD1 buildings and their influence on applications. In: Proceedings of the ISPRS 3D GeoInfo 2014 conference. Breunig, M. et al. (eds.). November 2014, Dubai, UAE.

BILJECKI, F., ARROYO OHORI, K. (2015a): Automatic semantic-preserving conversion between OBJ and CityGML. EurographicsWorkshop on Urban Data Modelling and Visualisation 2015, pp. 25-30.

BILJECKI, F., HEUVELINK, G. B. M., LEDOUX, H., & STOTER, J. (2015b). Propagation of positional error in 3D GIS: estimation of the solar irradiation of building roofs. International Journal of Geographical Information Science, 29(12):2269-2294. DOI: http://doi.org/10.1080/13658816.2015.1073292

BILJECKI, F., LEDOUX, H., STOTER, J., VOSSELMAN, G. (2016a): The variants of an LOD of a 3D building model and their influence on spatial analyses. ISPRS Journal of Photogrammetry and Remote Sensing, vol. 116, pp. 42-54.

BILJECKI, F., LEDOUX, H., & STOTER, J. (2016b): An improved LOD specification for 3D building models. Computers, Environment and Urban Systems, 59, s. 25–37.

BILJECKI, F., LEDOUX, H., & STOTER, J. (2016c). Generation of multi-LOD 3D city models in CityGML with the procedural modelling engine Random3Dcity. ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., IV-4/W1, 51–59. http://doi.org/10.5194/isprs-annals-IV-4-W1-51-2016

BILJECKI, F. (2017): Level of detail in 3D city models. PhD thesis, TU Delft, 353 s.

BOETERS, R., a kol. (2015): Automatically enhancing CityGML LOD2 models with a corresponding indoor geometry. International Journal of Geographical Information Science, 29 (12), 2248–2268.

DANTZIG, G. B. (1963): Linear Programming and Extensions. Princeton University Press, Princeton, NY, USA, str. 648.

DAMEN, J., KREVELD, M., SPAAN, B. (2008): High quality building generalization by extending the morphological operators. 11th ICA Workshop on Generalization and Multiple Representation, Montpellier, France, s. 1–12.

FORBERG, A. AND MAYER, H.(2003): Squaring and Scale-Space Based Generalization of 3D Building Data. 5th Workshop on Progress in Automated Map Generalization. Available online at: http://www.geo.unizh.ch/ICA/docs/paris2003/papers03.html.

GE, L. (2018): Generalization of LOD2 buildings with different roof structures. Journal of Spatial Science, 4, s. 19.

GRÖGER, G., KOLBE, T.H., HÄFELE, K. H., NAGEL, C. (2012): OGC City Geography Markup Language (CityGML) Encoding Standard. Version 2.0.0., Open Geospatial Consortium.

GRÖGER, G., PLÜMER, L. (2012): CityGML – Interoperable semantic 3D city models. ISPRS Journal of Photogrammetry and Remote Sensing, 71, s. 12–33.

GUERCKE, R., GÖTZELMANN, T., BRENNER, C., SESTER, M. (2011): Aggregation of LOD1 building models as an optimalization problem. IPRS Journal of Photogrammetry and Remote Sensing, vol. 66.

HAUNERT J. H. (2008): Aggregation in map generalization by combinatoral optimization. Ph.D. Thesis. Leibniz Universitat Hannover. In Reihne C, Heft 626 of Deutsche Geodatische Kommission

KADA, M., (2007): Scale-dependent simplification of 3D building models based on cell decomposition and primitive instancing. In: S. Winter, M. Duckham, L. Kulik, B. Kuipers, eds. Spatial Information Theory. Berlin, Germany. Springer Verlag, s. 222–237.

KADA, M., (2011): Aggregation of 3D buildings using a hybrid data approach. Cartography and Geographic Information Science, 38 (2), s. 153–160.

LIU, P., LI, C., A LI, F., (2017): Texture-cognition-based 3D building model generalization. ISPRS International Journal of Geo-Information, 6 (206), 1–19.

MATOUŠEK, J. (2006): Lineární programování – Úvod pro informatiky. Katedra aplikované matematiky. Matematicko-fyzikalní fakulta Univerzity Karlovy, s. 107. Dostupné z: https://iti.mff.cuni.cz/series/2006/311.pdf [cit. 2020-02-10].

MCMASTER R.B, SHE, AK. S. (1992): Generalization in Digital Cartography. Association of American Cartographers, Washington D.C.

MUSIALSKI, P., WONKA, P., ALIAGA, D. G., WIMMER, M., VAN GOOL, L. AND PURGATHOFER, W. (2013): A Survey of Urban Reconstruction. Computer Graphics Forum 32(6), pp. 146–177

PUTTERN, J., OOSTEROM, P. (1998): New results with generalised area partitionings. Proc. 8th Int. Symposium on Spatial Data Handling, s. 485–495.

SESTER, M. (2007): 3D Visualization and Generalization. Proceedings of the 51st Photogrammetric Week, Stuttgart, Germany, 3–7 September, s. 285–295.

SGALL, J. (2018): Optimalizační metody. Lineární programování a kombinatorická optimalizace. Dostupné z: https://iuuk.mff.cuni.cz/~sgall/vyuka/OPT/opt.pdf [cit. 2020-02-10].

STRACHOTA, P. (2015): Procedurální modelování. FJFI ČVUT v Praze. Dostupné z: http://saintpaul.fjfi.cvut.cz/base/sites/default/files/POGR/POGR2/06.proceduralni_modelovani.pdf [cit. 2019-09-25].

WARE, J.M., JONES, C.B., BUNDY, G.L. (1995): A triangulated spatial model for cartographic generalisation of areal objects. COSIT, s. 173–192.

XIE, J. a kol. (2012): Automatic simplification and visualization of 3D urban building models. International Journal of Applied Earth Observation and Geoinformation, 18, s. 222–231.

Seznam příloh

Příloha č. 1: Výsledky optimalizační úlohy, matice center a výšky agregátů těla a střešní části budovy

Matice center

		ID Budovy												
		A	B	C	D	E	F	G	H	I	J	K	L	M
ID centra agregátů	A	0	0	0	0	0	0	0	0	0	0	0	0	0
	B	0	0	0	0	0	0	0	0	0	0	0	0	0
	C	0	0	0	0	0	0	0	0	0	0	0	0	0
	D	0	0	0	1	0	0	0	0	0	0	0	0	0
	E	1	0	1	0	1	0	0	0	0	0	0	0	0
	F	0	0	0	0	0	1	0	0	0	0	0	0	0
	G	0	0	0	0	0	0	1	0	0	0	0	0	0
	H	0	0	0	0	0	0	0	1	0	0	0	0	0
	I	0	1	0	0	0	0	0	0	1	0	0	0	0
	J	0	0	0	0	0	0	0	0	0	1	1	0	1
	K	0	0	0	0	0	0	0	0	0	0	0	0	0
	L	0	0	0	0	0	0	0	0	0	0	0	1	0
	M	0	0	0	0	0	0	0	0	0	0	0	0	0

Výška centra agregátů - pro tělo agregátu

ID centra agregátů	A	B	C	D	E	F	G	H	I	J	K	L	M
Výška	0,00	0,00	0,00	6,58	6,10	6,98	9,99	9,00	10,29	3,33	0,00	9,09	0,00

Výška centra agregátů - pro střešní část agregátu

ID centra agregátů	A	B	C	D	E	F	G	H	I	J	K	L	M
Výška	0,00	0,00	0,00	3,25	2,80	2,65	3,55	2,31	2,66	2,80	0,00	2,85	0,00

Příloha č. 2: Výsledky optimalizační úlohy, matice objemových změn

Matice objemových změn těla budovy

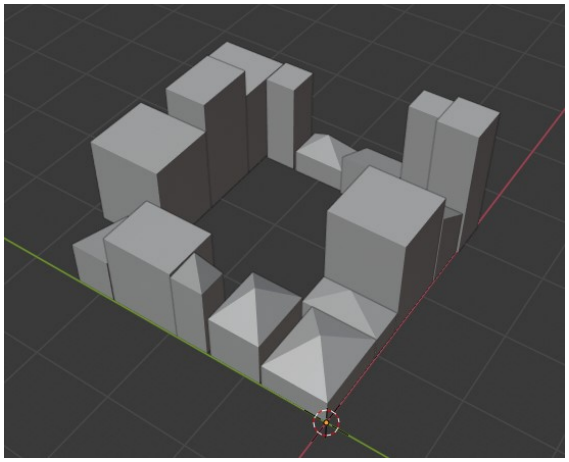
		ID Budovy												
		A	B	C	D	E	F	G	H	I	J	K	L	M
ID centra agregátů	A	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	B	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	C	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	D	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	E	56,06	0,00	224,85	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	F	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	G	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	H	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	I	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	32,00	0,00	0,00	0,00	0,00
	J	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	5,37	0,00	343,12
	K	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	L	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	M	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Matice objemových změn střešní části budovy

		ID Budovy												
		A	B	C	D	E	F	G	H	I	J	K	L	M
ID centra agregátů	A	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	B	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	C	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	D	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	E	9,18	0,00	0,00	0,00	9,14	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	F	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	G	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	H	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	I	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	7,11	0,00	0,00	0,00	0,00
	J	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	10,89
	K	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	L	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	M	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Příloha č. 3: Experimentální výsledky optimalizační úlohy: testovací model 2

Původní model bloku



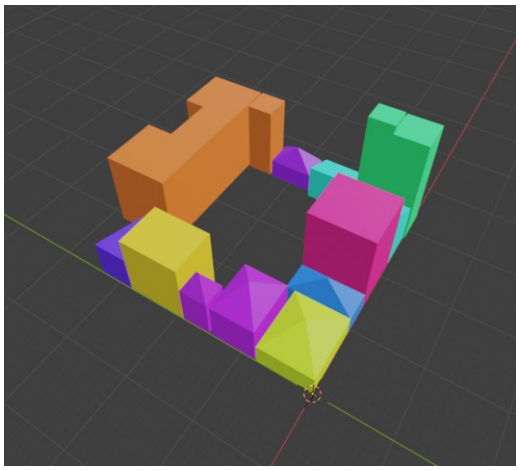
Původní model	
počet objektů	16
součet objemu budov v původním modelu	6848,34

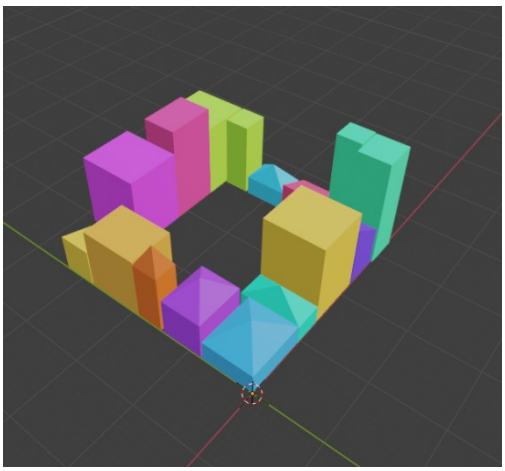
Nastavení parametru epsilon	
Parametr	Hodnota
epsilon_roof_type	0
epsilon_roof_orientation	1
epsilon_roof_height	5
epsilon_height	10

Výsledky optimalizace

Výsledky optimalizace – testovací model 2	
WB= 0,001 WR=0,001	
hodnota objektivní funkce	11,24
počet objektů	11
suma objemových změn těla budovy (abs)	239,31
suma objemových změn střešní části (abs)	1,33
suma objemových změn celkem (abs)	240,64
součet objemu celého modelu	6663,09

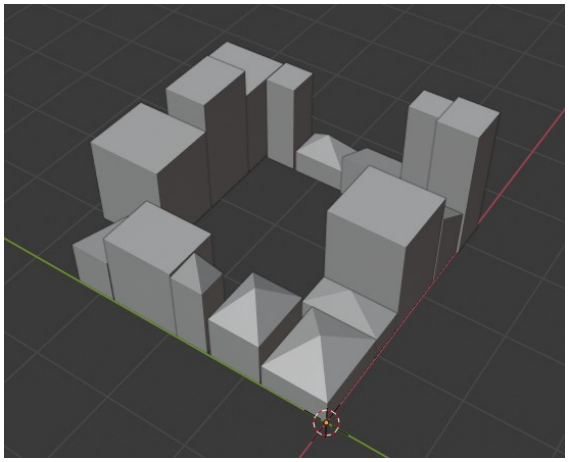
Výsledky optimalizace – testovací model 2	
WB= 0,05 WR=0,05	
hodnota objektivní funkce	14,94
počet objektů	14
suma objemových změn těla budovy (abs)	18,78
suma objemových změn střešní části (abs)	0,00
suma objemových změn celkem (abs)	18,78
součet objemu celého modelu	6857,76





Příloha č. 4: Experimentální výsledky optimalizační úlohy: testovací model 2 – extrémy

Původní model bloku



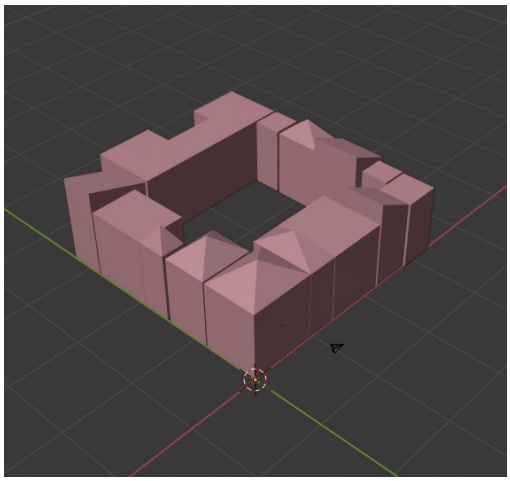
Původní model	
počet objektů	16
součet objemu budov v původním modelu	6848,34

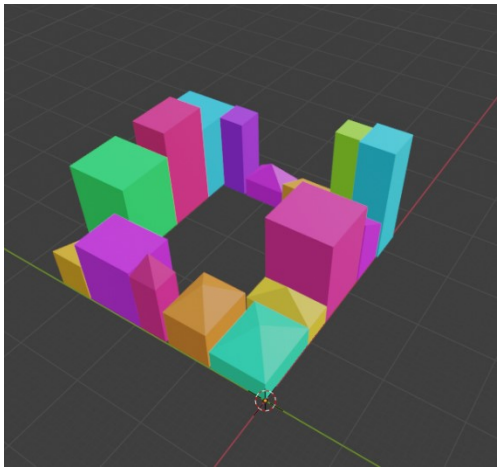
Nastavení parametru epsilon	
Parametr	Hodnota
epsilon_roof_type	5
epsilon_roof_orientation	360
epsilon_roof_height	10
epsilon_height	20

Výsledky optimalizace

Výsledky optimalizace – testovací model 2, extrémy	
WB= 0,0001 WR=0,0001	
hodnota objektivní funkce	1,31
počet objektů	1
suma objemových změn těla budovy (abs)	3040,42
suma objemových změn střešní části (abs)	21,58
suma objemových změn celkem (abs)	3062,00
součet objemu celého modelu	7516,63

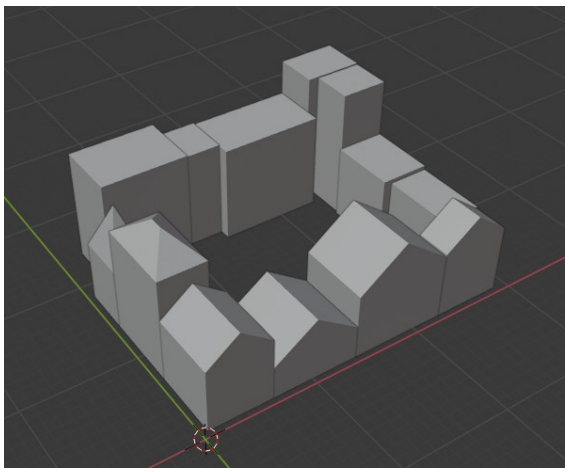
Výsledky optimalizace – testovací model 2, extrémy	
WB= 0,9 WR=0,9	
hodnota objektivní funkce	16,00
počet objektů	16
suma objemových změn těla budovy (abs)	4,98E-11
suma objemových změn střešní části (abs)	0,00
suma objemových změn celkem (abs)	4,98E-11
součet objemu celého modelu	6848,34





Příloha č. 5: Experimentální výsledky optimalizační úlohy: testovací model 3

Původní model bloku



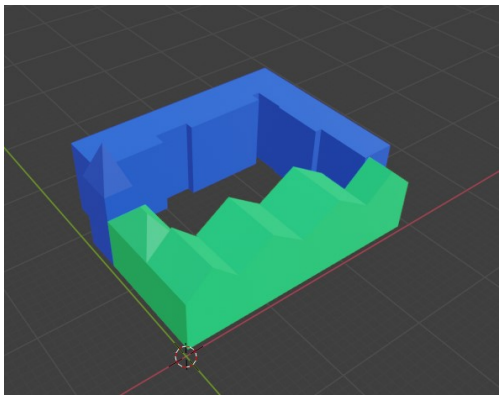
Původní model	
počet objektů	13
součet objemu budov v původním modelu	3362,51

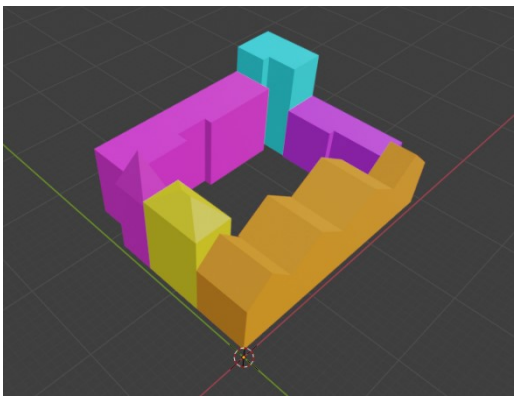
Nastavení parametru epsilon	
Parametr	Hodnota
epsilon_roof_type	2
epsilon_roof_orientation	0
epsilon_roof_height	5
epsilon_height	10

Výsledky optimalizace

Výsledky optimalizace – testovací model 3	
WB= 0,001 WR=0,001	
hodnota objektivní funkce	2,66
počet objektů	2
suma objemových změn těla budovy (abs)	621,54
suma objemových změn střešní části (abs)	35,18
suma objemových změn celkem (abs)	656,72
součet objemu celého modelu	3399,45

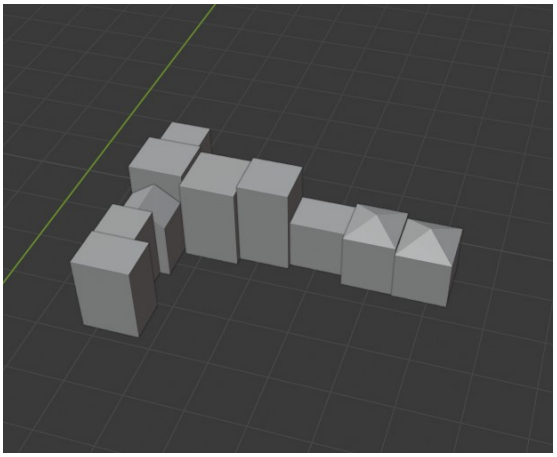
Výsledky optimalizace – testovací model 3	
WB= 0,01 WR=0,01	
hodnota objektivní funkce	8,10
počet objektů	5
suma objemových změn těla budovy (abs)	282,88
suma objemových změn střešní části (abs)	27,39
suma objemových změn celkem (abs)	310,27
součet objemu celého modelu	3488,80





Příloha č. 6: Experimentální výsledky optimalizační úlohy: testovací model 4

Původní model bloku



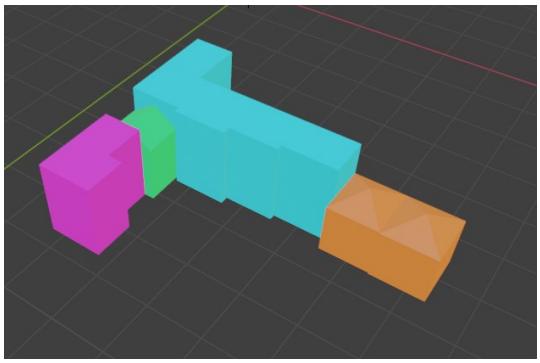
Původní model	
počet objektů	10
součet objemu budov v původním modelu	9924,85

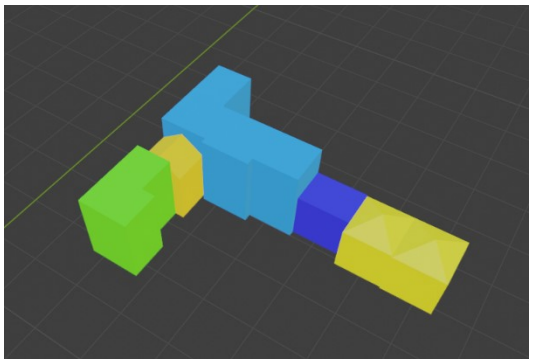
Nastavení parametru epsilon	
Parametr	Hodnota
epsilon_roof_type	0
epsilon_roof_orientation	0
epsilon_roof_height	5
epsilon_height	10

Výsledky optimalizace

Výsledky optimalizace – testovací model 4	
WB= 0,001 WR=0,001	
hodnota objektivní funkce	4,83
počet objektů	4
suma objemových změn těla budovy (abs)	812,64
suma objemových změn střešní části (abs)	13,64
suma objemových změn celkem (abs)	826,28
součet objemu celého modelu	10253,11

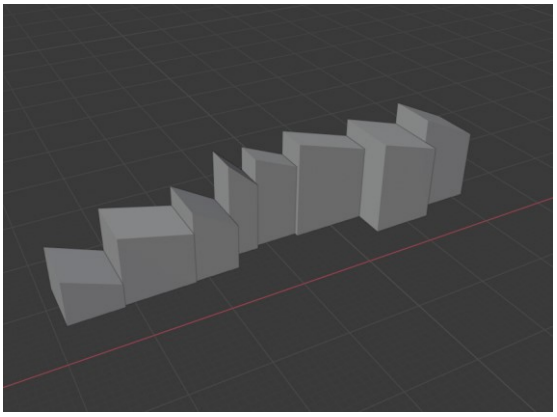
Výsledky optimalizace – testovací model 4	
WB= 0,005 WR=0,005	
hodnota objektivní funkce	7,12
počet objektů	5
suma objemových změn těla budovy (abs)	411,08
suma objemových změn střešní části (abs)	13,64
suma objemových změn celkem (abs)	424,72
součet objemu celého modelu	9851,55





Příloha č. 7: Experimentální výsledky optimalizační úlohy: testovací model 5

Původní model bloku



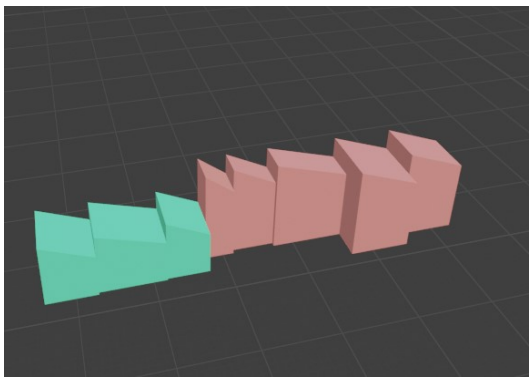
Původní model	
počet objektů	8
součet objemu budov v původním modelu	3286,91

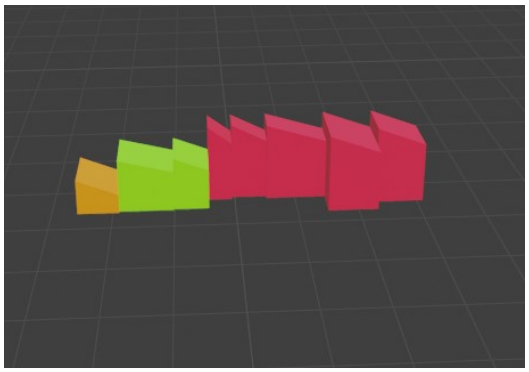
Nastavení parametru epsilon	
Parametr	Hodnota
epsilon_roof_type	0
epsilon_roof_orientation	0
epsilon_roof_height	2
epsilon_height	3

Výsledky optimalizace

Výsledky optimalizace – testovací model 5	
WB= 0,001 WR=0,001	
hodnota objektivní funkce	2,32
počet objektů	2
suma objemových změn těla budovy (abs)	247,63
suma objemových změn střešní části (abs)	76,58
suma objemových změn celkem (abs)	324,22
součet objemu celého modelu	3428,16

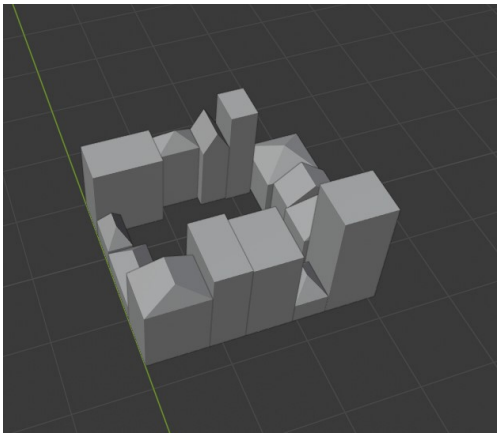
Výsledky optimalizace – testovací model 5	
WB= 0,01 WR=0,01	
hodnota objektivní funkce	4,59
počet objektů	3
suma objemových změn těla budovy (abs)	88,11
suma objemových změn střešní části (abs)	70,55
suma objemových změn celkem (abs)	158,66
součet objemu celého modelu	3341,19





Příloha č. 8: Experimentální výsledky optimalizační úlohy: testovací model 6

Původní model bloku



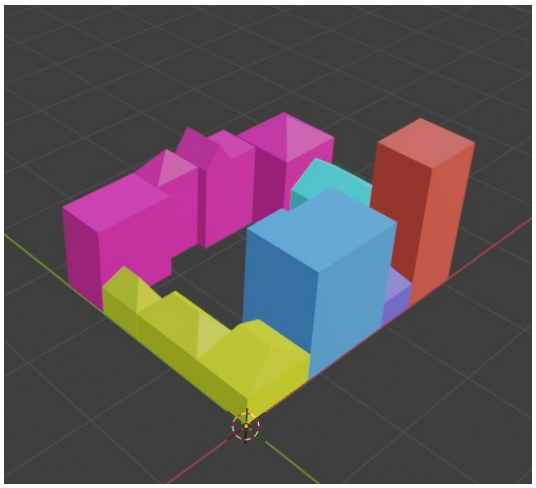
Původní model	
počet objektů	14
součet objemu budov v původním modelu	5520,31

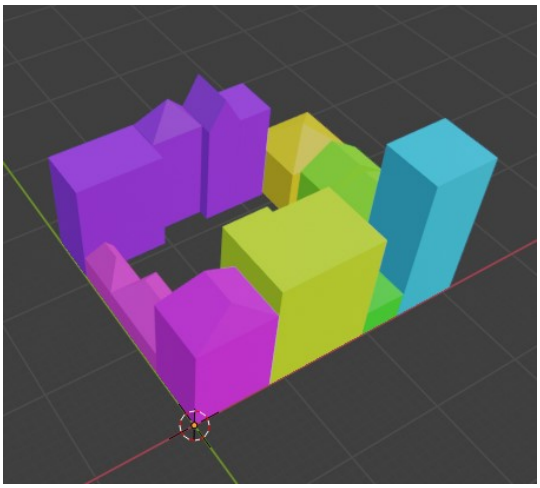
Nastavení parametru epsilon	
Parametr	Hodnota
epsilon_roof_type	2
epsilon_roof_orientation	180
epsilon_roof_height	5
epsilon_height	10

Výsledky optimalizace

Výsledky optimalizace – testovací model 6	
WB= 0,001 WR=0,001	
hodnota objektivní funkce	6,87
počet objektů	6
suma objemových změn těla budovy (abs)	852,40
suma objemových změn střešní části (abs)	21,70
suma objemových změn celkem (abs)	874,09
součet objemu celého modelu	5198,69

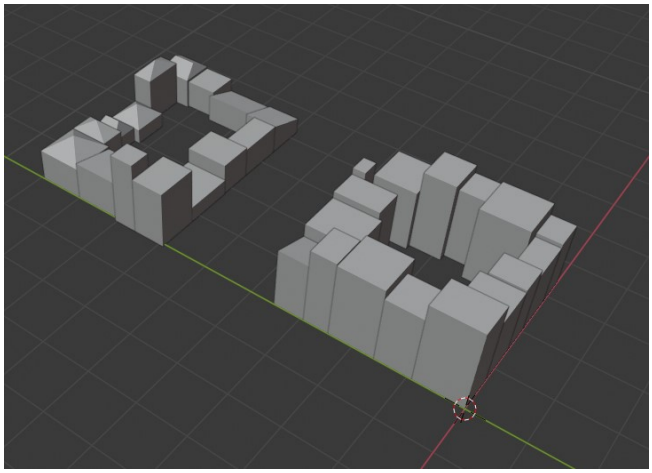
Výsledky optimalizace – testovací model 6	
WB= 0,01 WR=0,01	
hodnota objektivní funkce	9,72
počet objektů	8
suma objemových změn těla budovy (abs)	159,38
suma objemových změn střešní části (abs)	12,41
suma objemových změn celkem (abs)	171,79
součet objemu celého modelu	5497,06





Příloha č. 9: Experimentální výsledky optimalizační úlohy: testovací model 7 – dva bloky

Původní model bloku



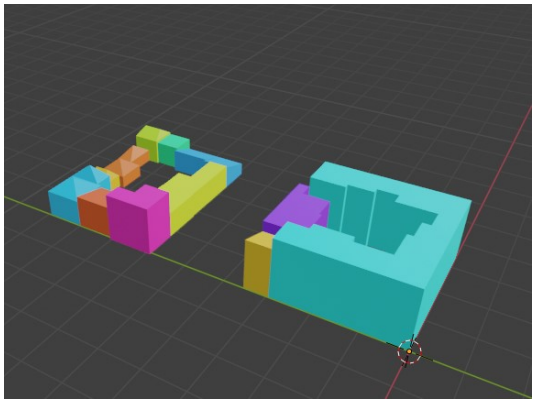
Původní model	
počet objektů	31
součet objemu budov v původním modelu	16580,85

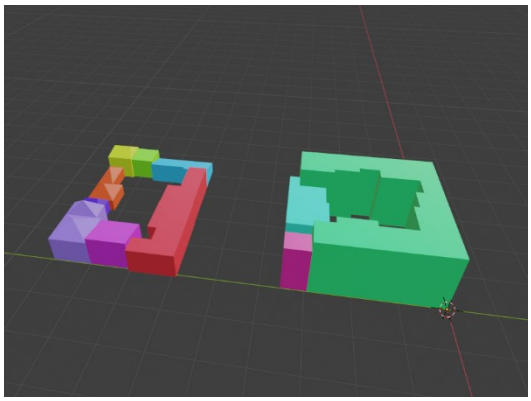
Nastavení parametru epsilon	
Parametr	Hodnota
epsilon_roof_type	0
epsilon_roof_orientation	0
epsilon_roof_height	rozdílné
epsilon_height	rozdílné

Výsledky optimalizace

Výsledky optimalizace – testovací model 7	
WB= 0,001 WR=0,001 epsilon_roof_height=4 epsilon_height=7	
hodnota objektivní funkce	13,52
počet objektů	12
suma objemových změn těla budovy (abs)	1501,14
suma objemových změn střešní části (abs)	14,44
suma objemových změn celkem (abs)	1515,58
součet objemu celého modelu	16640,70

Výsledky optimalizace – testovací model 7	
WB= 0,001 WR=0,001 epsilon_roof_height=5 epsilon_height=10	
hodnota objektivní funkce	12,96
počet objektů	11
suma objemových změn těla budovy (abs)	1946,26
suma objemových změn střešní části (abs)	14,44
suma objemových změn celkem (abs)	1960,69
součet objemu celého modelu	16330,82

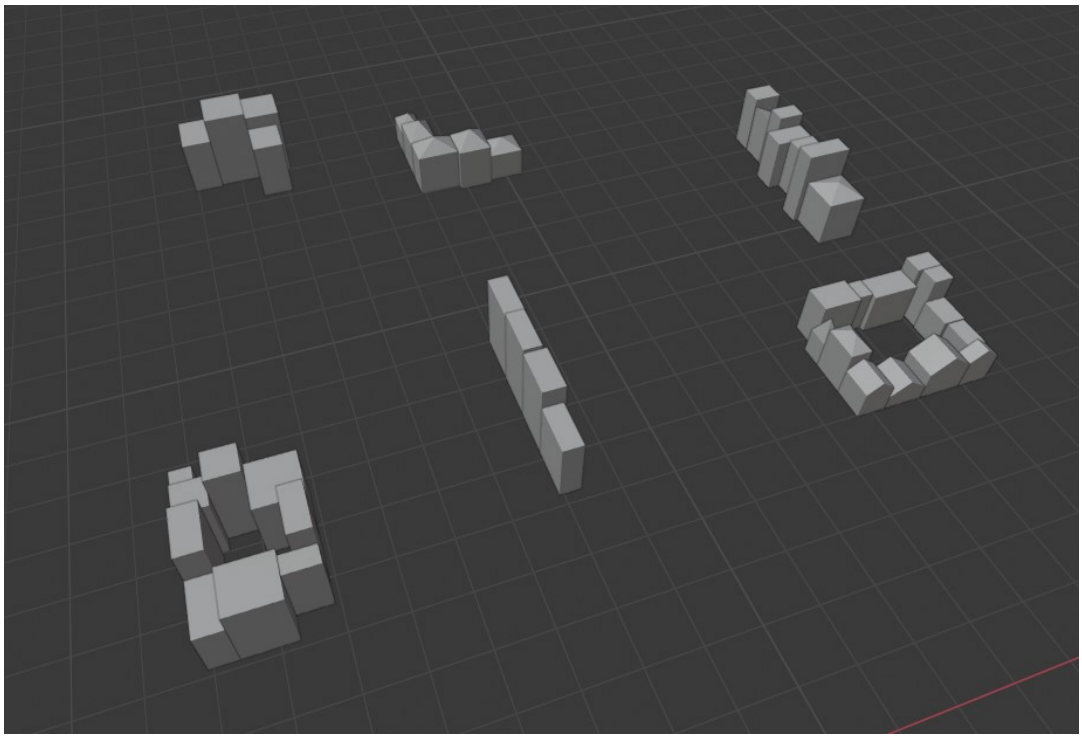




Příloha č. 10: Experimentální výsledky optimalizační úlohy: testovací model 8 – více bloků – první část

---

Původní model bloku



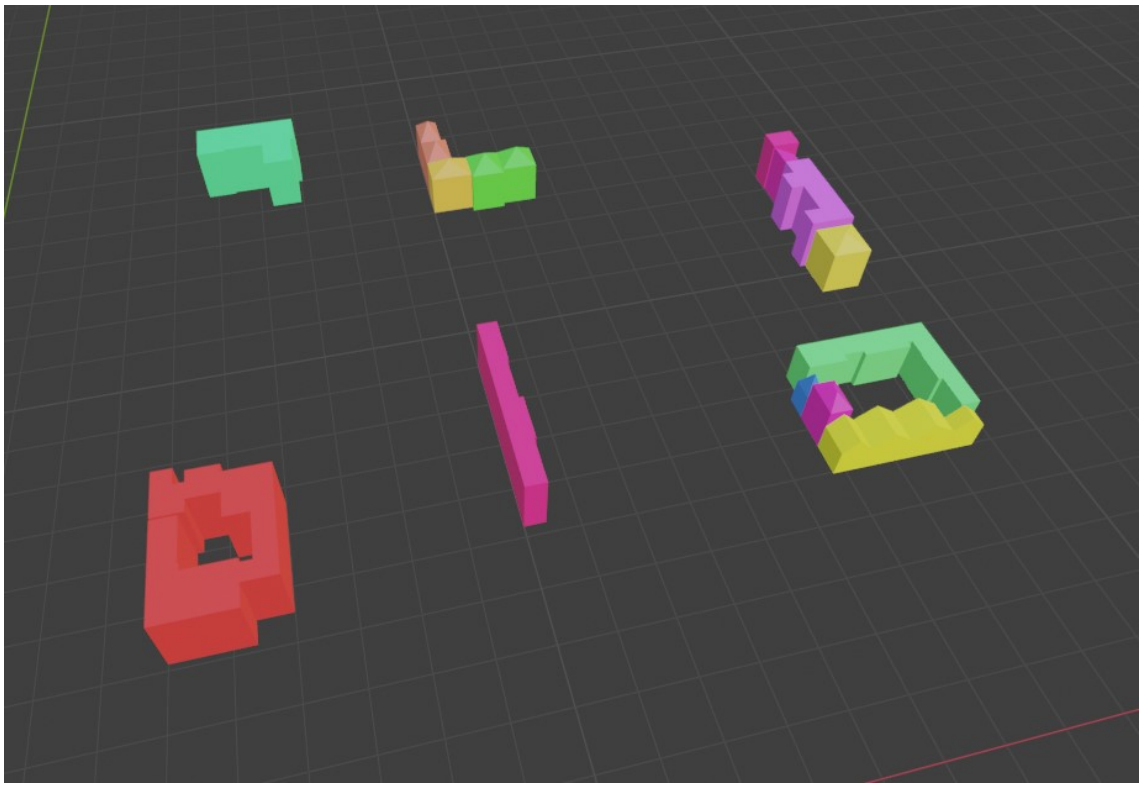
Původní model	
počet objektů	42
součet objemu budov v původním modelu	20980,59

Nastavení parametru epsilon	
Parametr	Hodnota
epsilon_roof_type	0
epsilon_roof_orientation	0
epsilon_roof_height	5
epsilon_height	10

Příloha č. 10: Experimentální výsledky optimalizační úlohy: testovací model 8 – více bloků – druhá část

---

Výsledky optimalizace



Výsledky optimalizace – testovací model 8	
WB= 0,001 WR=0,001	
hodnota objektivní funkce	16,99
počet objektů	14
suma objemových změn těla budovy (abs)	2943,08
suma objemových změn střešní části (abs)	49,23
suma objemových změn celkem (abs)	2992,31
součet objemu celého modelu	19506,75